UNIVERSITÉ DE MONTRÉAL

THEORETICAL AND APPLIED FOUNDATIONS FOR INTRUSION DETECTION IN SINGLE AND FEDERATED CLOUDS

ADEL ABUSITTA
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(GÉNIE INFORMATIQUE)
DÉCEMBRE 2018

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

THEORETICAL AND APPLIED FOUNDATIONS FOR INTRUSION DETECTION IN SINGLE AND FEDERATED CLOUDS

présentée par : ABUSITTA Adel
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. PIERRE Samuel, Ph. D., président
Mme BELLAÏCHE Martine, Ph. D., membre et directrice de recherche
M. DAGENAIS Michel, Ph. D., membre et codirecteur de recherche
M. QUINTERO Alejandro, Doctorat, membre
M. GRÉGOIRE Jean-Charles, Ph. D., membre externe

DEDICATION

To my wife Amira and our sons Ahmed and Saif,
for providing me love, support and inspiration....

ACKNOWLEDGMENTS

I would like to thank my Ph.D. supervisors Dr. Martine Bellaïche and Dr. Michel Dagenais for their understanding, continuous guidance, great supervision and full support. I was lucky to be surrounded by such professional, inspirational, and caring advisors. Thanks for believing in my potential and letting me explore the field of Cloud Computing and cyber-security. Without your guidance and persistent support, this dissertation would not have been possible.

Moreover, I would like to thank my Ph.D. committee members Professor Samuel Pierre, Professor Alejandro Quintero, and Professor Jean-Charles Grégoire for sparing their precious time in order to review and evaluate my thesis.

Furthermore, I would like to thank all my colleagues in the research lab at Polytechnique Montreal. In particular, I would like to thank my dear colleague and friend Talal Halabi for having created an inspiring and dynamic research environment.

I would like also to thank my friends in Canada and overseas. In particular, a special thanks to Omar Abdel Wahab for his help and support during my Ph.D studies in Canada. Your presence has allowed me to love my stay in Montreal and added lots of fun to my Ph.D. journey.

I would like also to thank Dr. Adel Serhani for his help and support during my Ph.D studies in Canada.

In addition, a special thanks to Dr. Naser Taleb for encouraging me to do my Ph.D in Canada.

I would like also to thank my friend Saad Tolba for his help and support during my Ph.D studies in Canada.

Also, I would like to thank my friend Mohammad Hleihel for his help and support during my stay in Montreal.

Last but not the least, I want to thank my parents, brothers and sisters for continuously supporting me throughout my Ph.D and for providing me with the opportunity to follow the right path for my future. Thank you for believing in me.

And finally, my warmest thanks to my wife Amira and our sons Ahmed and Saif, for dreaming with me, for being patient when I was busy and not available for them, and for providing me inspiration, support and love. Thank you for believing in me.

RÉSUMÉ

Les systèmes infonuagiques deviennent de plus en plus complexes, plus dynamiques et hétérogènes. Un tel environnement produit souvent des données complexes et bruitées, empêchant les systèmes de détection d'intrusion (IDS) de détecter des variantes d'attaques connues. Une seule intrusion ou une attaque dans un tel système hétérogène peut se présenter sous des formes différentes, logiquement mais non synthétiquement similaires. Les IDS traditionnels sont incapables d'identifier ces attaques, car ils sont conçus pour des infrastructures spécifiques et limitées. Par conséquent, une détection précise dans le nuage ne sera absolument pas identifiée. Outre le problème de l'infonuagique, les cyber-attaques sont de plus en plus sophistiquées et difficiles à détecter. Il est donc extrêmement compliqué pour un unique IDS d'un nuage de détecter toutes les attaques, en raison de leurs implications, et leurs connaissances limitées et insuffisantes de celles-ci.

Les solutions IDS actuelles de l'infonuagique résident dans le fait qu'elles ne tiennent pas compte des aspects dynamiques et hétérogènes de l'infonuagique. En outre, elles s'appuient fondamentalement sur les connaissances et l'expérience locales pour identifier les attaques et les modèles existants. Cela rend le nuage vulnérable aux attaques «Zero-Day». À cette fin, nous résolvons dans cette thèse deux défis associés à l'IDS de l'infonuagique : la détection des cyberattaques dans des environnements complexes, dynamiques et hétérogènes, et la détection des cyberattaques ayant des informations limitées et / ou incomplètes sur les intrusions et leurs conséquences. Dans cette thèse, nous sommes intéressés aux IDS génériques de l'infonuagique afin d'identifier les intrusions qui sont indépendantes de l'infrastructure utilisée. Par conséquent, à chaque fois qu'un pressentiment d'attaque est identifié, le système de détection d'intrusion doit être capable de reconnaître toutes les variantes d'une telle attaque, quelle que soit l'infrastructure utilisée. De plus, les IDS de l'infonuagique coopèrent et échangent des informations afin de faire bénéficier chacun des expertises des autres, pour identifier des modèles d'attaques inconnues.

L'originalité de cette thèse repose sur deux aspects : 1) la conception d'un IDS générique de l'infonuagique permettant la détection dans des environnements changeants et hétérogènes et 2) la conception d'un IDS coopératif multi-infonuagique garantissant fiabilité, équité et durabilité. Par « digne de confiance», nous entendons que l'IDS du nuage sera en mesure de s'assurer qu'il consultera, coopérera et partagera les connaissances avec des IDS de confiance d'autres nuages. Par équité, l'IDS de l'infonuagique sera en mesure de garantir que des avantages mutuels seront obtenus en minimisant les chances de coopération avec des IDS égoïstes.

Ceci est important pour motiver les IDS à participer à la communauté. Enfin, par durabilité on entend qu'un IDS d'un nuage prendra des décisions proactives en cas d'intrusion suspecte, même en l'absence d'information complète de feedback, ou de connaissances approfondie des IDS consultés. Ainsi, la solution proposée sera fiable et réalisable dans des environnements en temps réel, où les décisions relatives aux intrusions doivent être prises rapidement. Le travail dans cette thèse se déroule en trois phases.

Dans la première phase, nous proposons une structure permettant de surveiller et d'analyser les effets d'environnements hétérogènes et changeants (par exemple, l'adaptation et les ajustements de ressources) sur les données collectées et inspectées utilisées par les systèmes de l'infonuagique. La structure proposée filtre ces effets et supprime les données d'exécution non pertinentes de l'ensemble des données, afin de fournir des fonctionnalités robustes et génériques à celles-ci, améliorant ainsi la détection au niveau de toutes les infrastructures. Deux algorithmes sont proposés dans cette phase : l'algorithme d'analyse et l'algorithme de détection. L'algorithme d'analyse détermine les changements des données collectées et supprime les détails d'exécution non pertinents pour y améliorer la précision de l'algorithme de détection.

Au cours de la deuxième phase, nous proposons une structure de confiance et d'équité dans les systèmes de stockage intégrés multi-infonuagique. Pour calculer la confiance, chaque système IDS du nuage est doté d'une fonction de confiance permettant de calculer les valeurs de confiance d'autres systèmes IDS. En particulier, l'inférence bayésienne est utilisée pour calculer les valeurs de confiance des interactions précédentes. Par la suite, un nouvel algorithme décentralisé est conçu, basé sur la théorie des jeux de coalition, permettant aux IDS de l'infonuagique d'établir leurs coalitions pour maximiser la confiance des fédérations constituées, et augmenter la précision de détection individuelle en présence de IDS (malveillants ou non). Le modèle de coopération proposé, basé sur la confiance, converge vers une situation stable de Nash ; c'est-à-dire qu'aucun IDS de l'infonuagique n'a l'incitation de quitter sa coalition actuelle et d'en rejoindre une autre. Nous proposons également un algorithme d'agrégation de feedback basé sur la confiance pour y agréger les feedback reçus d'autres IDS de nuages de la même coalition. L'algorithme proposé d'agrégation a la propriété d'empêcher les attaques de collusion, qui se produisent lorsque plusieurs IDS en nuage collaborent pour envoyer des jugements trompeurs.

D'autre part, pour garantir l'équité, nous formulons un mécanisme de garantie d'équité basé sur un jeu de Stackelberg entre les IDS en nuage ayant un bon comportement et les égoïstes qui envoient fréquemment des demandes de consultation et ne répondent pas aux consultations des autres IDS, dans le but de sauver leurs propres ressources. Le mécanisme proposé

permet aux IDS performants de jouer la stratégie optimale minimisant les chances de coopération avec les égoïstes. De plus, nous proposons un problème d'optimisation basé sur un jeu de Stackelberg qui guide l'IDS d'un nuage pour déterminer la stratégie optimale (le taux de consultation et le taux de réponse) des activités déloyales. Les résultats théoriques et expérimentaux montrent que, selon notre modèle, les IDS en nuage n'ont aucune incitation à se comporter de manière égoïste.

Dans la dernière phase, nous concevons un IDS coopératif multi-infonuagique proactif en y intégrant des approches d'apprentissage automatique. La solution proposée exploite les données historiques pour prédire les intrusions suspectes. La prédiction est effectuée proactivement sans qu'il ne soit nécessaire d'appliquer une méthode d'agrégation (par exemple, la théorie de Dempster-Shafer ou le DST) sur le feedback d'informations des IDS consulté, ni d'attendre d'avoir reçu tous les commentaires des IDS consultés ; c'est-à-dire qu'un feedback d'informations partiel ou incomplet peut être utilisé. En particulier, le modèle proposé est basé sur un Denoising Autoencoder (DA), qui construit d'un réseau de neurones profonds. Les caractéristiques de DA nous permettent d'apprendre à reconstituer les informations des IDS à partir des feedback partiels ou incomplets, et d'extraire des fonctionnalités robustes et utiles pour prendre des décisions efficaces en cas d'intrusion suspecte, même en l'absence de feedback complet des IDS.

Nous concluons notre thèse en soulignant certaines lacunes de la recherche à approfondir à l'avenir.

ABSTRACT

Cloud Computing systems are becoming more and more complex, dynamic and heterogeneous. Such an environment frequently produces complex and noisy data that make Intrusion Detection Systems (IDSs) unable to detect unknown variants of known attacks. A single intrusion or an attack in such a heterogeneous system could take various forms that are logically but not synthetically similar. This, in turn, makes traditional IDSs unable to identify these attacks, since they are designed for specific and limited infrastructures. Therefore, the accuracy of the detection in the cloud will be very negatively affected. In addition to the problem of the cloud computing environment, cyber attacks are getting more sophisticated and harder to detect. Thus, it is becoming increasingly difficult for a single cloud-based IDS to detect all attacks, because of limited and incomplete knowledge about attacks and implications.

The problem of the existing cloud-based IDS solutions is that they overlook the dynamic and changing nature of the cloud. Moreover, they are fundamentally based on the local knowledge and experience to perform the classification of attacks and normal patterns. This renders the cloud vulnerable to "Zero-Day" attacks.

To this end, we address throughout this thesis two challenges associated with the cloud-based IDS which are: the detection of cyber attacks under complex, dynamic and heterogeneous environments; and the detection of cyber attacks under limited and/or incomplete information about intrusions and implications. We are interested in this thesis in allowing cloud-based IDSs to be generic, in order to identify intrusions regardless of the infrastructure used. Therefore, whenever an intrusion has been identified, an IDS should be able to recognize all the different structures of such an attack, regardless of the infrastructure that is being used. Moreover, we are interested in allowing cloud-based IDSs to cooperate and share knowledge with each other, in order to make them benefit from each other's expertise to cover unknown attack patterns. The originality of this thesis lies within two aspects: 1) the design of a generic cloud-based IDS that allows the detection under changing and heterogeneous environments and 2) the design of a multi-cloud cooperative IDS that ensures trustworthiness, fairness and sustainability. By trustworthiness, we mean that the cloud-based IDS should be able to ensure that it will consult, cooperate and share knowledge with trusted parties (i.e., cloud-based IDSs). By fairness, we mean that the cloud-based IDS should be able to guarantee that mutual benefits will be achieved through minimising the chance of cooperating with selfish IDSs. This is useful to give IDSs the motivation to participate in the community.

Finally, by sustainability, we mean enabling a cloud-based IDS to proactively make decisions about suspicious intrusions, even in the absence of complete feedback and knowledge from consulted IDSs. Thus, the proposed solution will be reliable and applicable in real-time environments, where decisions about intrusions need to be taken quickly. The work in this thesis is carried out in three phases.

In the first phase, we propose a framework that allows low-overhead monitoring, and analysing the effects of heterogeneous and changing environments (e.g., scaling and resources adjustments) on the collected and inspected data that are used by cloud-based IDSs. The proposed framework filters out these effects, and removes irrelevant run-time details from the data, in order to provide robust and generic features for the data, to enhance the detection for any possible infrastructure. Two algorithms are proposed in this phase: the analysis and detection algorithms. The analysis algorithm determines the changes that occur in the collected data, and removes irrelevant run-time details to enhance the accuracy of the detection algorithm.

In the second phase, we propose a framework for trust and fairness assurance in multi-cloud cooperative IDSs. To ensure trust, each cloud-based IDS is endowed with a belief function to compute trust values of other IDSs. In particular, Bayesian inference is used to compute trust values based on previous interactions. Thereafter, a novel decentralized algorithm is devised, based on the coalitional game theory, that allows cloud-based IDSs to establish their coalitions in such a way that maximises the trust of the formed federations, and makes individual detection accuracy increase, even in the presence of untrusted (malicious or not) IDSs. The proposed trust-based cooperative model converges to a Nash-stable situation; that is, no cloud-based IDS has an incentive to leave its current coalition and join another one. We also propose a trust-based feedback aggregation algorithm to aggregate feedbacks received from other cloud-based IDSs in the same coalition. The proposed aggregation algorithm has the property of preventing collusion attacks, which occur when several cloud-based IDSs collaborate to give misleading judgments.

On the other hand, to ensure fairness, we formulate a fairness-assurance mechanism based on a Stackelberg game between the well-behaving cloud-based IDSs and the selfish ones that frequently send consultation requests, and do not answer other IDSs consultations with the aim of saving their own resources. The proposed mechanism enables the well-behaving IDSs to play the optimal strategy that minimizes the chances of cooperating with the selfish ones. Moreover, we devise an optimization problem based on a Stackelberg game that guides the cloud-based IDS to determine the optimal strategy (the amount of consultation rate and response rate) as a response to unfair activities. The theoretical and experimental results

show that, according to our model, cloud-based IDSs have no incentive to behave selfishly.

In the last phase, we design a proactive multi-cloud cooperative IDS while integrating machine learning approaches. The proposed solution exploits the historical data to predict the status of suspicious intrusions. The prediction is done proactively, without the need to apply any aggregation method (e.g., Dempster-Shafer theory or DST) on consulted IDSs feedback, neither to wait until receiving all the feedback from the consulted IDSs, i.e., only partial or incomplete feedback can be used. In particular, the proposed model is based on a Denoising Autoencoder (DA), which is used as a building block for constructing a deep neural network. The characteristics of DA enable us to learn how to reconstruct IDSs feedbacks from partial or incomplete feedbacks and allow us to extract robust and useful features to make efficient decisions about suspicious intrusions, even in the absence of complete feedback from IDSs.

We conclude the thesis by highlighting some research gaps that require further investigation in the future.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

IT	Information Technology
ICT	Information and Communications Technology
AWS	Amazon Web Service
EC2	Elastic Cloud Compute
VM	Virtual Machine
SLA	Service Level Agreement
CPU	Central Processing Unit
QoS	Quality of Service
API	Application Programming Interface
DoS	Denial of Service
DDoS	Distributed Denial of Service
LTTng	Linux Trace Toolkit next generation
SDAE	Stacked Denoising Autoencoders
CP	Cloud Provider
IEEE	Institute of Electrical and Electronics Engineers
NDoS	Networked Denial of Service
XML	Extensible Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
DA	Denoising Autoencoder
CF	Cloud Federation
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
ICMP	Internet Control Message Protocol
SOAP	Simple Object Access Protocol
SF	Sensor Filter
ML	Machine Learning
AI	Artificial Intelligence
VB	Virtual Bridge
TSP	Time Spent on a Page
BSS	Blind Source Separation
FTA	Fault Tree Analysis
DST	Dempster-Shafer Theory

OS	Operating System
MTF	Multilevel Thrust Filtration
REST	Representational State Transfer
CBF	Confidence-Base Filtering
E-EMD	Ensemble Empirical Mode Decomposition
URL	Uniform Resource Locator
HC	Hierarchical Correlation
SCL	Secure Sockets Layer
SVM	Support Vector Machine
BPD	Beta Probability Density

CHAPTER 1 INTRODUCTION

In this chapter, we present the context of our research work, define the problems addressed in this thesis, indicate the corresponding research questions, and finally identify the objectives of our research work. The background and related work will be presented in the next chapter (Chapter 2).

1.1 Problem definition

Several major Information and Communications Technology (ICT) companies are competing for creating useful cloud computing services that are able to deal with different business requirements. In fact, many industries, companies, and governments are expected to transfer, if not already done, all or parts of their Information Technology (IT) solutions to the cloud [1] [2] [3]. This historical shift is profitable from the economic point of view since it enables them to streamline the spending on technology infrastructure and capital cost. However, this has led to growing concerns about the cyber attacks and security vulnerabilities that these complex systems might present.

With the existing complex, dynamic and heterogeneous architecture of the cloud, the ability of traditional Intrusion Detection Systems (IDSs) to compensate variant forms of the same attack was dramatically affected. A cloud system consists of different types of operating systems such as Linux and Windows machines. Moreover, a cloud computing system is largely dynamic and heterogeneous, due to the inherent characteristics and changing environments (e.g., resources restriction, migration and scaling), which are essential to meet the requirements of the pay-as-you-go business model [4]. A single intrusion in such an environment can have variant forms that are semantically but not synthetically similar. These lead cloud-based IDSs, whether they are signature or anomaly-based approaches, to deal with irrelevant runtime details and complexity instead of focusing on essential information used to increase the detection accuracy.

To this end, a generic IDS is required to deal with all variants of attacks. In fact, achieving a generic IDS, that is able to compensate for different structures of the same attack, is challenging. This is due to the fact that it requires extra overhead in order to monitor every change in the cloud and understand to which extent the collected data have been affected by the new infrastructure. Although one can manually define higher-level features and abstract data details to come up with a generic IDS, this approach is difficult, since it requires the

availability of experts in the knowledge domain, in order to understand many details and complex information. Such experts cannot easily be found for clouds that are largely public, dynamic and multi-tenant.

Another problem, as a result of the complex architecture of the cloud, is that cyber attacks have evolved, getting more sophisticated and harder to detect. Thus, it is becoming very hard for a single cloud-based IDS to detect all existing attacks, due to limited knowledge about such attack patterns and implications.

While several approaches have been proposed to model the cooperation among IDSs (e.g. [5] [6] [7] [8] [9] [10]) to solve the problem of limited information, these approaches, however, work under the assumption that all cloud-based IDSs are trustable. This makes their collaboration systems vulnerable to untrusted (malicious or not) insiders. Moreover, the lack of trustworthy framework in the cooperative setting leads to collusion attacks, which occur when several malicious cloud-based IDSs collaborate to give misleading judgments.

Although achieving trustworthiness in cloud-based cooperative IDSs could enhance the coalition members detection performance, the motivation of the cloud-based IDSs to participate in the detection process needs further investigation. An IDS can be trustworthy but selfish at the same time. A selfish IDS is one that frequently sends consultation requests to other IDSs and, at the same time, does not answer other IDSs consultation requests, with the aim of saving its own resources. Therefore, the fairness property of the cooperation model needs to be incorporated in the framework so as to encourage the cooperation and provide suitable incentives.

Moreover, there are considerable delays associated with the existing cooperative IDSs. These delays are mostly due to the computation complexity of using the aggregation algorithms (e.g., Dempster-Shafer theory), and also due to the large geographic distances that might separate the cloud-based IDSs. In fact, each IDS, after receiving feedback from consulted IDSs about a suspicious intrusion, is required to use a suitable feedback algorithm, in order to make a final decision about the suspicious intrusion. The aggregation technique is usually costly in terms of computation time, and depends on many factors, such as the number of consulted IDSs, and the IDSs expertise and trust levels [11] [12]. Also, due to uneven IDSs connections and Internet speeds, and other unknown factors (e.g, busy IDSs, compromised IDSs), the feedbacks are not guaranteed to be received at the same time. Thus, decisions on whether to raise an alarm about suspicious intrusions or not, might be unduly delayed due to the missing feedback of a single IDS. Hence, the decisions generated by the cooperative IDS are ineffective in a real-time setting, making them unsustainable.

All of the above problems have led to the rise of the following research questions that we

addressed thoroughly throughout the thesis :

— How to design and build a low overhead monitoring and analysis system in order to understand the effect of cloud heterogeneity and changing environment on the collected data ?
— How to integrate these effects into the existing or new detection algorithms in order to enhance the detection accuracy in the cloud ?
— How to enable a cloud-based IDS to evaluate another IDSs trustworthiness, in order to allow trustworthy IDS communities ?
— How to exploit trust information on clouds and IDSs to establish trust-based federated clouds and IDSs ?
— How to aggregate IDSs feedback inside cloud-based IDS communities in such a way as to prevent collusion attacks ?
— How to formulate a fairness assurance mechanism between well-behaving IDSs and selfish ones in order to enable the well-behaving IDS to play the optimal strategy and thus reduce the chances of cooperating with selfish IDSs ?
— How to enable proactive multi-cloud cooperative IDSs in real-time environments so that an IDS can proactively make decisions about suspicious intrusions, even in the absence of complete feedback from the IDSs ?

1.2 Research objectives

This thesis essentially aims at enabling and enhancing the detection of cyber attacks in the cloud under complex, dynamic and heterogeneous environments, and under limited or incomplete information about intrusions. More specifically, the objectives of the thesis are :

— Proposing a detection approach to identify various forms of an intrusion in the cloud under complex, changing and heterogeneous environments.
— Proposing a low overhead monitoring and analysis framework to understand the effect of heterogeneous and changing environments on the collected data in the cloud.
— Proposing a trust-based evaluation approach that enables a cloud-based IDS to evaluate another IDS trustworthiness, based on its past experiences.
— Modeling and proposing a framework that enables cloud-based IDSs to distributively form trustworthy cloud-based IDS communities.
— Devising an algorithm for forming coalitions among cloud-based IDS which leads to stability i.e, the case where none of the coalition members has an incentive to leave its current coalition and join another one.
— Proposing a new trust-based aggregation approach that enables the aggregation of

cloud-based IDSs feedbacks belonging the same community, in such a way as to prevent collusion attacks.

— Formulating a fairness assurance mechanism between the well-behaving IDSs and the selfish ones.

— Proposing a proactive multi-cloud cooperative IDS that enables us to make decisions about suspicious intrusions even with partial IDSs feedback, in order to accelerate the decision making in real-time environments.

— Designing a proactive multi-cloud cooperative IDS that is able to make decisions about suspicious intrusions without the need to apply aggregation methods on IDSs feedback.

1.3 Main contributions and their originality

The main originality of this thesis lies in the design of models and algorithms that 1) allow and enhance the detection in Cloud Computing systems, which are becoming more complex, dynamic and heterogeneous, and 2) enable trustworthy, fair and proactive multi-cloud cooperative IDS. The proposed IDS can be applied and localized at any level in single and federated clouds including, for examples, it can be localized at the networks, hypervisors, operating systems, and/or VMs levels. The principal contributions can be described as follows :

— **Proposing a generic cloud-based IDS to detect intrusions or attacks under complex, changing and heterogeneous environments.** We design a framework that enables a low-overhead monitoring and analysis of changes (e.g., resource scaling) in the cloud in order to understand the effects of these changes on the collected data used by an IDS. We propose an algorithm that helps us filter out these effects and remove irrelevant run-time details from the collected data in order to provide robust and useful features that are then integrated into the proposed detection algorithm. The proposed solution can take as input any kind of data with respect to the OSI model [13]. For example, in the data link layer, it can take frames as inputs ; in the network layer, it can take packets as inputs ; and in the transport layer, it can take segments as inputs. It is worth mentioned here that the proposed model can reduce the overhead associated with the monitor and analysis of large data because of the following two methods that are adopted in our approach. First, we adopt hooks strategy, which can be used and applied at any given layer. This strategy enables an IDS to monitor the load of latency and trigger more monitoring and analysing processes only when a problem is suspected, not when the system is largely idle. Second, since our model is based on distributed algorithms, this allows the overhead to be distributed among several nodes instead of having a single point of failure.

— **Proposing a trust-based multi-cloud cooperative IDS.** We introduce a trust framework that enables a cloud-based IDS to evaluate other cloud-based IDS trustworthiness, based on Bayesian inference [14]. Thereafter, a hedonic game theoretical model [15] combined with the Bayesian inference is proposed, to establish trustworthy cloud-based IDS communities. The proposed framework ensures Nash stability. The proposed model also enables a trust-based coalition to be formed among Cloud Providers (CPs), in order to allow a CP to outsource some of its workload to other CPs. This serves a dual purpose, reducing the extra overhead (monitoring and detection) in cloud-based IDSs, and exploiting the power of other IDSs in dealing with sophisticated and severe attacks.

— **Proposing a trust-based aggregation method for preventing collusion attacks in the federated cloud-based IDS.** We elaborate a Dempster-Shafer Theory (DST)-based model [16] which enables a cloud-based IDS to prevent collusion attacks, which occur when several malicious IDSs collaborate to give misleading judgments. The proposed model dramatically decreases the chances of considering these feedbacks that come from IDSs with low-level trust values during feedback aggregation. Moreover, it allows us to deal with uncertainty in which a cloud-based IDS has no information about a suspicious intrusion.

— **Proposing a fairness-assurance mechanism in multi-cloud cooperative IDS environments.** We design a fairness-assurance mechanism, which is based on the Stackelberg game [17] between the well-behaving cloud-based IDSs and the selfish ones, that frequently send consultation requests and do not answer other IDSs consultations with the aim of saving their own resources. The proposed model enables a cloud-based IDS to play the optimal strategy that minimizes the chances of cooperating with selfish IDSs.

— **Designing a proactive multi-cloud cooperative IDS.** We propose a machine learning-based cooperative IDS that efficiently exploits the historical feedback data in order to provide the ability of proactive decision making. The proposed model is based on a Denoising Autoencoder [18][19], which is used as a building block for constructing a deep neural network, which allows us to proactively make decisions about suspicious intrusions, even in the absence of complete feedback from the IDSs. In proactive multi-cloud cooperative IDS, it is important to identify the time period for sending and receiving information among IDSs. When an IDS has explored a suspicious intrusion and wants to consult other IDSs about that, the IDS should set a time period T so that it can make a decision proactively only after the time has expired. In fact, the value of T should be set probably by taking into consideration many factors such as :

the communication speed and geographic distances among IDSs. For this purpose, the proposed approach enables each IDS to set the value of T based on its own experience with other IDSs and also based on its own communication overhead.

1.4 Thesis structure

Chapter 2 will review the literature related to each element of the research problem that we described. An analysis of the limitations of existing work and the gaps that must be filled is addressed. In chapter 3, a detailed description of our research work and published articles is given, and the relationship between our objectives is emphasized.

Chapter 4 presents the full text of the article titled “An SVM-based Framework for Detecting DoS Attacks in Virtualized Clouds under Changing Environments”, which was published in the Journal of Cloud Computing.

Chapter 5 and chapter 6 present the full texts of the article titled “A Trust-based Game Theoretical Model for Cooperative Intrusion Detection in Multi-cloud Environments” and the article titled “On Trustworthy Federated Clouds : A Coalitional Game Approach” which were published in the "Conference on Innovation in Clouds, Internet and Networks" and "Computer Networks", respectively.

Chapter 7 presents the full text of the article titled “Multi-cloud Cooperative Intrusion Detection System : Trust and Fairness Assurance” which is submitted in “Annals of Telecommunications” Journal.

Chapter 8 presents the full text of the article titled “A Deep Learning Approach for Proactive Multi-Cloud Cooperative Intrusion Detection System” which is submitted to the “Future Generation Computer Systems”.

Chapter 9 presents a general discussion regarding the thesis strong points and limitations. Finally, chapter 10 concludes the thesis by presenting a summary of our contributions and a discussion on some research gaps that require further investigation in the future.

CHAPTER 2 BACKGROUND AND LITERATURE REVIEW

In this chapter, we discuss recent works that have been done in the areas connected to this thesis. We first introduce some basic definitions about the topic and then discuss the related works.

2.1 Definitions and basic concepts

This section aims at defining the terminology and concepts that will be used in the rest of the thesis.

2.1.1 Cloud Computing

Cloud Computing enables Cloud Providers (CPs) to rent out space on their infrastructures, platforms and services to many consumers. This becomes possible thanks to the virtualization that enables the easy migration of applications and services from one node to another. Many companies, organizations and governments are expected to transfer, if they have not already, all or parts of their IT solutions to the cloud [4]. This transfer is profitable from an economic point of view since it allows them to streamline technology infrastructure expenses and capital costs.

2.1.2 Cloud Federation

One of the main issues that must be faced by CPs, due to the huge demands on their services, is the problem of insufficient resources to fulfill the requested VMs. This motivates the need for CPs to delegate these requests to other CPs in order to upgrade their resource scaling capabilities. A Cloud Federation (CF) provides an effective platform to address the aforementioned challenges [20]. The purpose of the CF consists of grouping CPs to fulfill the dynamic resource requests of users/applications to support data-intensive workloads [21]. Thus, through the use of CFs, CPs can benefit from each other's resources to run the VMs [22], in order to improve individual performance and enhance users satisfaction.

2.1.3 Intrusion Detection System

Intrusion detection is the process of monitoring performance metrics in order to explore attack symptoms. Such symptoms might be either at an early stage, or advanced enough to

cause a significant performance degradation. There are two main types of IDSs : signature-based and anomaly-based [1]. The former compares suspicious behavior with known attack patterns. In order to make signature-based systems effective, the signature database should be updated frequently. On the other hand, anomaly-based IDSs raise alarms when unusual and/or unexpected activities are detected. Anomaly-based IDSs are effective in detecting unknown attacks. Moreover, they do not need a database of known attacks. IDSs may adopt both techniques to have an improved detection accuracy.

2.1.4 Cooperative Intrusion Detection System

It is becoming increasingly difficult for a traditional single intrusion detection system (IDS) to detect all attacks, due to limited knowledge about attacks. A collaboration among IDSs has proven its efficiency in terms of the accuracy in detecting new and sophisticated attacks [12] [11] [23]. Through collaboration, IDSs in different regions, and possibly belonging to different Cloud Providers (CPs), can cooperate in such a way to utilize the expertise of each other to cover unknown threat patterns. This can be done by enabling IDSs to consult each other about suspicious behavior, where the received feedback can then be used to decide whether to raise an alarm or not.

2.2 Cloud-based Intrusion Detection Systems

In this section, we classify the existing cloud-based attack detection approaches into three major categories : signature-based, anomaly-based and hybrid detection approaches.

2.2.1 Signature-based Detection

The signature based detection techniques employ a set of predefined rules and signature attack patterns stored in a certain database to compare them against the incoming and outgoing traffic patterns. For example, Lonea et al. [24] propose SNORT, a signature-based technique that is configured with predefined DoS rules in order to detect known DoS attacks in the cloud environment. They tested their method by simulating ICMP flood, TCP SYN and UDP flooding attacks using Stacheldraht [25]. Similarly, Bakshi and Yogesh [26] use SNORT on each VM, where SNORT is deployed at the virtual interface. This allows VMs to analyse in-bound and out-bound traffic in real-time. Snort-based detection techniques have the advantage of sniffing attacker packets and detecting Networked DoS (NDoS) attacks. However, it has a disadvantage that it cannot be used to extract performance at other layers than the communication layer, such as processes and threads.

Gupta and Kumar [27] use an attack pattern detection approach based on VMs profile optimization. They employ a rule-based detection method for matching packets during flooding attacks (e.g., TCP SYN) by generating, through the initial rule establishment, a threshold for each rule pattern. In a similar work, Gul and Hussain [28] integrate multi-threading on top of their approach to improve the detection performance. The main advantage of this approach is that it is able to detect DoS at the level of VM. However, if the VM is compromised, the detection accuracy rate might be affected.

In another related work proposed to detect application-based flooding attacks, Karnwal et al. [29], [30] propose an approach to detect XML-based and HTTP-based DoS attacks, which occur during the SOAP-based requests for resources. They use Deterministic Packet Marking (FDPM), an IP traceback system, to find and mark the real source of SOAP messages. Finally, they use five different filter stages to detect the attacks, which are : Sensor Filter (SF), Hop count, IP Frequency Divergence, Confirm legitimate user IP and double signature. The first four components are used to identify the HTTP-based DOS attacks while the fifth one is used to detect the XML-based DoS attacks. While the different stages of detection increase the attack detection rate, it is associated with a significant overhead that comes from the different and multiple stages.

Conclusive remarks. Although signature-based detection approaches provide high accuracy in detecting known attack patterns, the weakness of these approaches is that they are unable to identify unknown attack patterns. Recent attackers are becoming more expert to launch attacks in such a way that they cannot be detected from predefined patterns. Instead, sophisticated techniques could be adopted, benefiting from the complex nature of the cloud environment. Moreover, most of the recent tools used to launch DoS attacks are easy to get from the Internet and are able to create new types of attacks [31].

2.2.2 Anomaly-based Detection

This approach is more reliable than the signature-based approach in terms of its ability to detect unknown attack patterns. The goal of this approach is to distinguish the anomalies behavior from the expected behavior. We classify the state-of-theart anomaly-based detection techniques into two main categories : Machine Learning (ML) or Artificial Intelligence (AI) approaches, and Statistical approaches.

2.2.2.1 Machine Learning and Artificial Intelligence Approaches

This approach benefits from the advanced AI and ML techniques to improve the accuracy in detecting the DoS attacks in the cloud. For example, Lonea et al. [24] use the normal traffic pattern received from the Virtual Bridge (VB) of the VM to validate for consistency against behavioral patterns of attacks. They use a network intrusion detection system, that analyzes the normal traffic flow obtained from the VB, to check and test for consistency against the attack behavioral patterns. Thus, if abnormal traffic has been detected, the anomaly information will be reported and an alarm generated.

Similarly, Gupta et al. [27] propose a profile based network intrusion detection system. They combine both fine grained data analysis and Bayesian techniques in order to detect TCP SYN flooding. The main advantage of these approaches is the ability to identify DoS symptoms at an early stage, because their approaches are able to collect information at the networking level, before the DoS causes a significant performance degradation. However, the lack of application performance information can result in wrongly identifying high traffic, during the peak time, as a DoS attack.

Ficco et al. [32] propose a strategy, for generating stealthy DoS attacks in the cloud, which uses low overhead attacks to inflict the maximum financial cost to the cloud clients [32]. Masood et al. [33] propose a web-behavior-based detection, where they identify two client profiles. The first one is for good clients while the second one is for bad clients. A good client will follow a pattern that reflects normal activity on the web, while a bad client will show some abnormal activities. Similarly, Anusha et al. [34] study the behavior of normal users of Web applications. They assume that an attacker spends a very short time (almost zero) over a Web page. They use for that a metric called Time Spent on a Page (TSP). They assume that the attackers TSP is very close to zero. In contrast, the TSP of a normal client should be high enough to interact with the Web page. The work of Kwon et al. [35] also uses a behavioral approach for detection. They start from a tested assumption saying that the behavioral patterns of normal traffic are similar, while the behavior patterns of malicious traffic are not. The cosine similarity is used to check the similarity of the traffic. If such a similarity does not exist, an alarm is generated. A main advantage of this approach that it is able to determine the similarity during run-time. However, there is no guarantee that the normal traffic will always be similar in a dynamic environment (i.e. cloud). In fact, in some applications, we could have many forms of normal traffic that are fairly dissimilar.

Palmieri et al. [36] use a two-phase ML-based detection technique. The first phase is called Blind Source Separation (BSS), while the second phase is called Rule-based Classifier to detect zero-day attacks that change or alter the traffic volume rate. BSS extracts the features of the

cloud nodes traffic, in order to be used by a decision tree classifier to create a normal traffic profile (baseline). Most recently, Choi et al. [37] propose a data-mining-based approach to detect application layer HTTP GET DoS attacks. They use a normal behavior pattern to detect DoS attacks on VMs. The parameters used for analyzing and creating attack patterns are : CPU usage, packet size and packet header information. They evaluate their approach by comparing it with a signature-based approach. The result showed that their proposed method performs better than SNORT in terms of identifying new attack profiles. Similar to this work, Jeyanthi and Mogankumar [38] and Jeyanthi et al. [39] propose a mechanism to detect DDoS attacks based on clients request rates. Clients requests will be put in a black list or white list based on a certain threshold rate. The threshold is determined by calculating the maximum number of legitimate client requests. The authors have shown experimentally that the legitimate clients could continue being served during an attack using their method. However, the main disadvantage of this method is that it is threshold based, where setting the optimal threshold is always difficult and possibly infeasible in a production cloud environment.

Chonka and Abawajy [40] and Chonka et al. [41] propose a decision tree classification technique. The method operates in two phases : training phase (first phase) and testing phase (second phase). In the training phase, a rule set that has been generated over time by the decision tree classifier is used to define both known and unknown attributes. In the testing phase, a decision making module is used to decide the likelihood of a previously classified packet. This helps decide whether to let a packet enter or not. Similarly, Lonea et al. [24] also proposed a classification technique based on Intrusion detection system (IDS). The detection module analyses the alerts generated by each VM using the Dempster-Shafer theory (quantitative solution classifier) in 3-valued logic and fault tree analysis (FTA). Although Dempster-Shafer is able to produce powerful results, when observations about attacks come from different sources, it becomes unsuitable when one source produces multiple observations [42].

Among other approaches, the work of Iyengare et al. [43] proposes a Multilevel Thrust Filtration (MTF) that contains four detection and prevention modules, which are traffic analysis, anomaly detection, anomaly classification, and attack prevention. The proposed method filters the incoming packets and detects four types of traffic congestion, which are spoofing, ash crowd, DDoS, and aggressive legitimate traffic. A similar approach has been proposed by Jeyanthi and Iyengar [39]. The main feature of this approach is the ability to increase the attack detection accuracy because multiple stages of detection are used. However, it is associated with a significant overhead since multiple algorithms and techniques should be used.

Recent attackers are able to exploit the vulnerability in the Representational State Transfer (REST) API. The vulnerability of REST results from the fact that REST does not require authentication. This lets cloud-based services to be overloaded. Michelin et al. [44] propose a mechanism to detect such attacks. They use two different modes ; monitoring and filtering. During the monitoring phase, a stress test is performed on the system to check for any overload. Thereafter, A token from each user is verified. If an invalid token is detected, the user is considered as a potential attacker by registering him/her in a gray list. When an overload on the REST API occurs, the mode will be changed from monitoring to filtering and the gray list will become black. Users in the black list will have their REST access inactivated. The main advantage of their work is identifying a new vulnerability that could be exploited by the attackers. However, the stress test that is required in this approach is time consuming, and therefore is not feasible in production environments.

2.2.2.2 Statistical Approaches

Statistics-based detection is based on calculating statistical features of a normal traffic to generate a normal traffic pattern. The normal traffic pattern will then be compared with the incoming traffic to detect malicious packets. For instance, Dou et al. [45] propose a statistical approach based on Confidence-Base Filtering (CBF). The concept of CBF reflects how much trust can be put on a correlation characteristic between attributes [46]. The CBF use a nominal profile during non-attack periods to extract statistical features from the cloud system resources. The occurrence frequency of these features is used to build a confidence value. During the attack period (run-time environment), the statistical features of cloud resources is compared with the nominal profile to identify the abnormal traffic, which will be considered as an actual attack. Similarly, Negi et al. [47] propose an enhanced CBF packet filtering method based on [45] to improve the utilization and processing time of the storage based on correlating patterns. A statistical approach was also used to detect application-based DoS attacks in the cloud. For example, Shamsolmoali and Zareapoor [48] applied two levels of filtering to detect DoS attacks in a cloud. The first level removes the IP header packet and compares the TTL value with the stored value. If these values are dissimilar, the packet is dropped and marked as spoofed. In the second level, they use the Jensen-Shannon divergence concept [49] that employs a stored normal profile to compare the incoming packet header data and check for data divergence. The statistical information used to detect DoS attacks is reliable because it is based on the CBF, which is the main advantage of this approach. However, the disadvantage of this approach is that it requires non-attack periods, which requires disconnecting applications and services, to extract statistical features from the cloud system resources.

Among other statistical approaches, Vissers et al. [50] propose a technique to detect malicious XML code contained in a SOAP message. To this end, they use the Gaussian distribution [51] to model the normal profile. To generate a normal distribution, the following two steps are used. In the first step, the file having all datasets is retrieved and all the entries that belong to a particular SOAP action are grouped together in smaller datasets. In the second step, a Gaussian model is generated by determining the means and standard deviations. A Gaussian distribution is also used recently by Marnerides et al. [52] who propose the Ensemble Empirical Mode Decomposition (E-EMD), which is an anomaly detection framework. The E-EMD can be used at the hypervisor level to collect statistical network information from VMs. An average and standard deviation is calculated for each VM and this information is used to detect anomalies caused by DoS attacks. The Gaussian based approach has an advantage, it is able to identify the normal behavior of the system. However, using the Gaussian distribution is the simplest approach to model the normal behavior. In most cases, other approaches (e.g., functional approach) are required along with the statistical approaches in order to understand the normal behavior of a complex system.

The work of Ismail et al. [53] uses a covariance matrix to detect flooding attacks. They used an approach that consists of two phases. The first phase is for profiling normal traffic by mapping the captured or received normal traffic into a matching covariance matrix. The second phase is the detection phase, where the obtained covariance matrix is compared with the currently received traffic. Similarly, Girma et al. [54] recently combined covariance matrices and entropy-based systems to identify patterns of DoS attacks in VMs. The results have shown superiority in terms of detection rate compared to other related statistical approaches. Also, the work of Zakarya [55] enhances the detection rate by using an entropy rate to identify the DoS attack flow, based on the distribution ratio. Similarly, Bedi and Shiva [56] use an entropy approach to detect DoS attacks generated from the co-residents. They suggest to model the behavior of both malicious and legitimate clients. While entropy-based approaches could achieve high detection rates, the main disadvantage of such approaches is that they assume that the normal traffic is always higher than the malicious traffic. This assumption might be possible only if we consider some simple DoS attack scenarios. However, in a cloud environment, we could have a large number of compromised VMs and the malicious traffic might be higher than the normal traffic.

Conclusive remarks. The existing anomaly-based detection techniques perform better than the signature-based detection techniques in terms of detecting unknown attack patterns. However, the anomaly-based detection techniques often require significant computing resources to identify the normal behavior of VMs (e.g., traffic rate). In addition, the existing

detection work lacks support for adaptation, with the dynamic adjustments that characterise the cloud computing environment which are important to meet the quality of service requirements and pay-as-you-go business model. In other words, the performance information that is used for learning attack patterns is becoming inappropriate for the new adjustments on the cloud infrastructure.

2.2.3 Hybrid detection

This approach is used to achieve higher detection rate by using some complementary features from the signature-based detection approaches and other features from the anomaly-based detection approaches. For example, Modi et al. [57] propose an approach that combines SNORT and Bayesian classifiers. They use SNORT to store the rules of known patterns of DoS attacks. The Bayesian classifier is used to predict the probability that system events belong to either the normal or malicious classes. The work of Cha and Kim [58] proposes a detection approach consisting of three phases. The first phase is monitoring and uses a rule-based technique to process DoS attack patterns.The second phase predicts the future load of each customer using timeseries modeling. They use a Bayesian technique to analyse DoS attack candidates on the network. The last phase uses an unsupervised learning algorithm [59] (the authors did not mention which unsupervised learning algorithm they used) to detect known and also unknown DoS attack patterns. The main advantage of using the Bayesian approach is that it provides a solid theoretical framework for classifying and detecting DoS attacks. However, it needs a big dataset in order to make reliable estimations of the probability for normal or malicious classes.

The work of Ficco [60] proposes a hybrid hierarchical correlated approach. It consists of detecting DoS attacks symptoms by collecting diverse information at several cloud architectural levels (e.g., application, VM). In order to identify attack patterns, a Hierarchical Correlation (HC) approach is used. HC captures the causal relationships between the alarms, which were generated through the previous stage (intermediate attacks in a complex attack), by correlating them on the base of temporal and logical constraints. The correlation capability is driven by a knowledge-based represented by an ontology. The correlation capability is then used to capture the causal relationship between the detected intermediate DoS attacks. The main advantage of this approach is the ability to detect sophisticated DoS attacks in cloud. The reason for that is because it uses causal relationships between intermediate attacks that belong to the sophisticated attack. However, the disadvantage of this approach is that it assumes that the causal relationships are built from true alarms. In fact, these alarms might be false positives and could decrease the attack detection accuracy.

Teng et al. [61] propose an approach that combines two detectors : feature detector and statistical detector. The feature detector uses SNORT to separate events based on network protocols (e.g., TCP). The Statistical detector cooperates with the feature detector by using data packets from it to determine whether an event is an attack or not. If the rate of obtained packets exceeds the predefined threshold, then this case will be considered as an attack. This approach can provide a high accuracy in detecting known networking-based DoS attack patterns. However, the weakness of these approaches is that they are unable to identify unknown attack patterns. Moreover, setting the optimal threshold is commonly difficult and infeasible in a changing environment (i.e., cloud).

Conclusive remarks. The existing work that use hybrid techniques have the advantages of combining signature and anomaly approaches. However, these approaches are unable to meet the cloud computing inherent characteristics. The reason is that they inherit the shortcomings of both signature-based and anomaly-based approaches. In addition, the hybrid techniques are associated with a significant overhead that comes from the different and multiple algorithms that are being used.

2.3 Cloud-based Cooperative Intrusion Detection System

In this section, we present the state-of-the-art of the cloud-based cooperative IDSs. Cooperative IDSs in the context of cloud computing have been proposed in many earlier works. For example, Lo et al. [62] propose a cooperative detection method in the virtualized cloud environment. Their method allows alerts to be exchanged among different nodes (i.e., hosts) whenever an attack gets detected. For this purpose, they adopt a rule-based technique to identify TCP SYN attacks by fetching the threshold for rule patterns during the initial rule establishment phase. The advantage of this method is that it is able to balance the detection overhead among nodes. Also Teng et al. [61] proposed a method that aggregates two types of detectors : a feature detector and a statistical detector. The former uses SNORT to separate events based on network protocols (e.g., TCP). The later cooperates with the feature detector by using data packets from it to decide whether an event is an attack or not. If the rate (i.e., the rate of packets) obtained is greater than the predefined threshold, then this situation will be considered as an attack.

Man and Huh [63] and Singh et al. [64] propose a cooperative IDS between cloud computing regions. Their approaches enable exchanging alerts from multiple elementary detectors. In addition, they enable sharing information between interconnected clouds. Also, Ghribi [65] proposed a middleware IDS. The approach allows a cooperation between three layers :

Hybervisor-based IDS, Network-based IDS and VM-based IDS. If an attack is found in a layer, the attack cannot be executed in the other layers. Chiba et al. [66] propose a cooperative network-based cooperative intrusion detection system to detect network attacks in the cloud. This can be performed through traffic monitoring while maintaining performance and service/application quality.

The main shortcoming of the above works is that they consider that all cloud-based IDSs are trustable, which lets their collaboration systems more vulnerable to untrusted and/or malicious insiders. The goal of this thesis is to present a systematic approach to establish a cloud-based cooperative IDS that uses trust assessment mechanisms and enables trustworthy decisions aggregation. We aim to allow our approach to work in the presence of untrusted and/or malicious IDSs .

In a multi-cloud environment, Dermott et al. [67] propose a cooperative intrusion detection in a federated virtualized cloud. They adopt the Dempster-Shafer theory of evidence to gather the beliefs provided by the watching entities. The gathered beliefs are used to reach the final decision regarding a possible attack. The main shortcoming of this approach is that it is a centralized architecture, whereby a trusted third-party should collect and manage feedbacks.

Cooperative IDSs in non-cloud environments where also proposed recently, in [68] [69] [5] [6] [7] [8] [9] [10]. They have the same shortcoming as the above mentioned works, since they assume that all IDSs are trustable, which makes their collaboration system vulnerable to malicious insiders.

A trust-based cooperative IDS has been proposed in a non-cloud environment in [11]. They propose a trust-based collaborative decision framework. Through collaboration, a native IDS can identify new attacks that may be known to other IDSs. The work evaluates how to use different diagnosis coming from different IDSs. They propose a system architecture for a collaborative IDS where trustworthy feedback aggregation is a key component. Similarly, Zhu et al. [70] [71] propose an incentive-based communication protocol, which gives IDS nodes incentives to share their feedback, and thus to prevent untrusted behaviors.

A fairness assurance mechanism in a cooperative IDS also has been proposed in [72] and [73]. They create a rule dissemination protocol based on a decentralized two-level optimization framework, which determines the information propagation rates to each IDS. For this purpose, they adopt a Bayesian learning approach for the IDS to find the compatibility ratio of other IDSs based on the historical interactions gathered by each IDS. The main limitation of their approach is that it is limited to signature-based cooperative IDSs, where the fairness is measured in terms of the ability to distribute rules fairly among IDSs.

2.4 Cloud Federation

In this section, we present the state-of-the-art of the cloud federation. The concept of federations among CPs was first introduced by Rochwerger et al. [20]. Although their work shows the main materials needed to achieve federation, they did not show the architectural elements that compose multi-cloud computing environments. Buyya et al. [74] introduce the challenges and architectural elements for federations. Similarly, Celesti et al. [21] and Fazio et al. [75] present a cloud architecture that allows CPs to build a federation with each other. They consider two kinds of CPs : home and foreign. The home CPs are those that are unable to fulfill the consumers tasks and therefore forward these jobs to the foreign CPs. Similarly, Goiri et al. [76] present a decision-based model that helps a CP decide on forming federations with public CPs in order to maximize their individual profit.

Toosi et al. [77] present multi-resource provisioning policies, that assist the CPs to increase their resource utilization and profit. Their model can terminate VMs whenever the profit of shutting them down exceeds the profit of running such VMs. Also, Van den Bossche et al. [78] present a binary integer program model that reduces the cost of outsourcing, using a mix of public and private providers. Chaisiri et al. [79] propose an optimal VM provisioning algorithm using stochastic programming that considers multi-cloud providers with the objective of maximizing their profit. Similarly, Bruneo [80] proposes a performance evaluation approach based on stochastic reward nets for federated CPs. The model predicts and quantifies the cost-benefit of a strategy portfolio and the corresponding QoS experienced by clients.

A business-oriented cloud federation model for real-time applications is proposed by Xiaoyu et al. [81]. The model allows multiple heterogeneous CPs to cooperate and provide a scalable infrastructure. The advantage is the business layer added to support the federations. The layer can trigger on-demand resource provisioning across multiple CPs and therefore helps to maximize the clients satisfaction and business benefits [81].

Salama & Shawish [82] present a QoS-based approach for cloud federation. They use QoS metrics such as throughput and response time during the federation formation process. By considering QoS metrics, the federation helps eliminate Service Level Agreement (SLA) violations and maximise QoS targets. In [83], Mashayekhy et al. propose a hedonic coalitional game to achieve cooperation among IaaS services. Based on the federation coalition game, they design a cloud federation formation mechanism that allows CPs to form federations that maximize their profits.

A game theoretic approach for cloud federation is also proposed by Hassan et al. [84]. The

study enables the dynamic resource allocation in a cloud federation. They define a price function for a CP that gives incentives to other CPs to contribute their resources in order to form a federation. Similarly, Mihailescu & Teo [85] present a strategy-proof dynamic pricing scheme for cloud federations. In [86], Li et al. propose profit maximization strategies in cloud federations. They present a truthful auction-based mechanism for selling VMs within a federation. This enables cloud federation members to sell or buy resources in a way that maximises their profit. Also, Samaan [87] proposes an economic model for sharing resources among CPs in the federation.

Few studies have addressed trust issues in cloud federations. For example, Ngo et al. [88] present an approach for attribute-based trust establishment to be used in the multi-cloud environment. They propose an approach for trust evaluation and delegation. Messina et al. [89] suggest a trust model based on the reputation. The model allows users to properly select a suitable CP on the basis of reliability and reputation.

Hassan et al. [90] propose a trust-based hedonic game to form a coalition among CPs. They enable CPs to join a coalition based on maximization of profits and minimization of penalty costs. The main limitation of their approach is that it is based on a centralized architecture, and a trusted third-party is required in order to organize the coalition. In [91], Wahab et al. designed a trust-based hedonic game to the model the community formation problem among multi-provider services. The main advantages of this approach lie in the (1) trust-based aggregation technique that can overcome collusion attacks in the presence of dishonest parties [91], (2) distributed trust-based coalition formation model that does not need a centralized entity, and (3) bootstrapping mechanism that assigns initial trust values for newly deployed services. The main limitations of this approach are that it considers functionally-similar services to create coalitions. Moreover, in this approach, untrusted services are considered as those that show some malicious behavior, whereas in some cases some untrusted services might be non-malicious (e.g., lack of experience). Besides, the computations of the trust values in this approach are limited to recommendations collected from different parties without considering self-experience.

2.5 Literature review analysis

In this section, we analyzed the limitations of the presented works and discussed the research gaps that need to be filled. First, although the aforementioned works paved the way to understand the issues behind improving the detection accuracy in cloud environments, these works overlook the heterogeneity and changing nature of the cloud. In fact, a single intrusion in such a heterogeneous environment can take various forms that are semantically but

not synthetically similar. These lead the existing approaches, whether they are signature or anomaly-based, to be unable to deal with irrelevant run-time details, instead of focusing on essential information used to increase the accuracy of the detection. In this thesis, we address these challenges by proposing a generic cloud-based IDS, in order to enable the detection of intrusions, regardless of the infrastructure being used.

Secondly, in the context of cloud-based cooperative IDS, while there are several cooperative models proposed to address the problem of limited information, by allowing to share knowledge among cloud-based IDS, they work under the assumption that all cloud-based IDSs are trustable. This, in turn, makes their collaboration systems vulnerable to untrusted (malicious or not) insiders. We address this point by proposing a trust-based multi-cloud cooperative IDSs. Moreover, despite the fact that some works (e.g., [11]) propose trust-based cooperative IDS, the main shortcoming of these works is that they are based on consulting many IDSs in order to get a feedback. This, in turn, causes extra overhead, through consulting needlessly some IDSs. This is unlike our approach, where a trust-based coalitional game approach is proposed, in order to construct a set of the most trusted and efficient IDSs, and thus reduce the rate of consultation requests, while ensuring a higher detection accuracy. In addition, the proposed framework has the following two advantages that are not available in the existing works : fairness assurance and sutainability. In the former, we design a fairness-assurance mechanism, based on the Stackelberg game [17] between the well-behaving cloud-based IDSs and the selfish ones that frequently send consultation requests and do not answer other IDSs consultation requests, with the aim of saving their own resources. In the latter, we devise a machine learning-based multi-cloud cooperative IDS that efficiently exploits the historical feedback data in order to provide the ability of proactive decision making. This makes our model reliable in real-time environments

Third, although some of the proposed approaches use the concept of cloud federation to exploit the power of other clouds in handling sophisticated and severe attacks, these approaches often suffer from the hazard of choosing unreliable and untrusted CPs in the federation. As a result, this leads to performance degradation and loss of CPs' reputation. We address this challenge by proposing a trust-based cloud federation. Although there are some works (e.g., [91]) that propose trust-based approaches to model the community formation problem among multi-provider services, the limitations of these approaches are that they consider functionally-similar services to create coalitions. Moreover, in their approaches, untrusted services are considered as those that show some malicious behavior, whereas in some cases some untrusted services might be non-malicious (e.g., lack of experience). Besides, the computation of the trust values in this approach are limited to recommendations collected from different parties, without considering self-experience.

CHAPTER 3 RESEARCH METHODOLOGY

The aim of this thesis is to propose a framework for a cloud-based IDS that can efficiently and effectively 1) detect attacks under complex, dynamic and heterogeneous environments in order to come up with a generic cloud-based IDS, and 2) detect unknown attacks about which a cloud-based IDS has no (or limited) information. To this end, several objectives were envisioned as announced in section 1.2. To attain these objectives, the work was carried out in three main phases. This chapter aims to explain the different steps that led to the realization of these objectives and connecting them to the work presented in the subsequent chapters.

3.1 Phase 1 : Intrusion detection under complex and changing environments

In the first phase of the thesis, our efforts were directed towards the achievement of the first two objectives. This phase consists of : 1) monitoring and analysing the effects of changing and heterogeneous environments on the collected data, in order to remove irrelevant run-time details and enhance the accuracy of the detection. ; 2) integrating the output of the proposed monitoring and analysis algorithm into the proposed detection algorithm, to come up with a generic cloud-based IDS. It is worth mentioning that we consider Denial of Service (DoS) attacks as a case study to achieve these objectives, since it is identified to be the most popular and dangerous attack [1] [2]. The above mentioned objectives were attained in the article titled “An SVM-based Framework for Detecting DoS Attacks in Virtualized Clouds under Changing Environment”.

3.1.1 Monitoring and analysing the effects of changing and heterogeneous environments

We first monitor data used for the detection, in order to measure the effects of the changing environment on this data. This was achieved by analysing and determining to which extent the collected data had been affected with respect to the applied changes (e.g., granting/revoking resources to/from the VMs). The monitoring and collection of data was performed using the Linux Trace Toolkit next generation (LTTng) [92], which enables a low-overhead monitoring and provides precise and detailed information on the underlying kernel and userspace executions [93]. We designed an algorithm to determine the changes that occurred in the data with respect to the changes that are applied to the cloud infrastructure. The output of the algorithm is a filter containing the effects of all possible changes on the data. The filter will

then be considered by the proposed detection algorithm, as we will see in the next section, to enhance the detection under changing and heterogeneous environments. Details are given in Chapter 4.

3.1.2 A generic cloud-based IDS

We proposed an SVM-based detection algorithm for this purpose. The SVM classifier was trained to distinguish between the normal and malicious activities. During the prediction period, the proposed algorithm first monitors and gathers data using LTTng. The filter obtained is used before applying the prediction using SVM, to get rid of the "noise" that may show up on the collected data (e.g., due to the new resource adjustments) and that considerably decreases the accuracy of the detection. The filtering process enables us to remove this noise and the irrelevant run-time details from the data, in order to provide a robust and generic dataset that will be used to enhance the detection in dynamic environments. Details about the proposed generic cloud-based IDS are given in Chapter 4.

3.1.3 Evaluation

To evaluate our model, we chose to create a custom test environment. All machines used in the experiments were attached directly to a Linksys 1000 Mb/s SOHO switch. Our test network was completely disconnected from the network of our institution, as well as from the Internet, to avoid the leakage of the DoS attacks. The proposed detection algorithm was implemented in Python and the BoNeSi program was used [94] to generate attack-level and normal traffic. BoNeSi allows us to simulate floods from large-scale bot networks. Moreover, BoNeSi tries to avoid the generation of packets with easily identifiable patterns, which can be quickly filtered out [94].

The experiments were done while applying different changing environments, including granting/revoking resources to/from the VMs. The results show that the proposed framework enhances the detection accuracy, (false positive and false negative), compared to other detection algorithms. Details about the experiments and results are given in Chapter 4.

3.2 Phase 2 : Trust and fairness assurance in multi-cloud cooperative IDSs

In the second phase, our efforts were directed towards the achievement of objectives 3,4,5 and 6. The work consists of : 1) trust-based multi-cloud cooperative IDSs, 2) trust-based feedback aggregation, 3) trust-based cloud federation, and 4) fairness-assurance in multi-cloud cooperative IDSs. The above mentioned objectives were attained in the following three articles :

“A Trust-based Game Theoretical Model for Cooperative Intrusion Detection in Multi-cloud Environments”, “On Trustworthy Federated Clouds : A Coalitional Game Approach”, and “Multi-cloud Cooperative Intrusion Detection System : Trust and Fairness Assurance”

3.2.1 Trust-based multi-cloud cooperative IDSs

To achieve trustworthy multi-cloud cooperative IDSs, we propose a framework for forming and establishing a trust-based coalition among cloud-based IDSs.

3.2.1.1 Proposed solution

We proposed a trust-based framework for cooperative IDS in a multi-cloud environment. The proposed model enables a cloud-based IDS to evaluate other IDSs trustworthiness, based on its experience, using Bayesian inference [14]. After obtaining IDSs trust values, a novel community formation algorithm is used. The proposed community formation algorithm is based on the coalitional game theory [95], [15]. The algorithm enables cloud-based IDSs to join or leave a given community so as to enhance their chances of working with trusted IDSs. The proposed algorithm enables each cloud-based IDS to discover trusted IDSs, and to list them on its whitelist. Our solation converges to a Nash-stable situation ; that is, no cloud-based IDS has an incentive to leave its current coalition to move to another coalition. Details are available in Chapter 5.

3.2.1.2 Evaluation

We implemented our framework in a 64-bit Windows 8 environment on a host equipped with an Intel Core i7-4790 CPU 3.60 GHz Processor and 16 GB RAM. We used Matlab for implementing our model. The simulation environment used 100 cloud-based IDSs. The results show that the detection accuracy was enhanced when the trust values of IDSs increased. Moreover, the results show that the proposed model dramatically reduces the chances of cooperating with untrusted (malicious or not) cloud-based IDSs. This, in turn, leads to both enhancing the accuracy and minimizing the cost of the cooperation compared to the "Grand cooperative approach", where the cooperation is conducted with all cloud-based IDSs. The reason is that a cloud-based IDS, in our model, is not required to cooperate with all other IDSs. Only trusted IDSs are considered. Details about the experiments and results are available in Chapter 5.

3.2.2 Trust-based Feedback Aggregation

In the previous section, we designed a trust-based community formation model that enables a set of cloud-based IDSs to cooperatively set up their coalitions. The output of the proposed model is a set of coalitions, where each coalition consists of a set of cloud-based IDSs that prefer to work with each other. In this section, we show how an IDS inside a coalition can aggregate feedback received from other cloud-based IDSs in the same community, in order to make a final decision about the suspicious intrusion.

3.2.2.1 Proposed solution

For this purpose, we proposed an aggregation algorithm based on the Dempster-Shafer Theory (DST) [16]. The advantages of using DST can be summerized as follows : (1 unlike other aggregation models (e.g. Bayesian aggregation model) that demand complete information of prior probabilities, DST can handle the lack of complete information (i.e. uncertainty), and (2 it has the property of preventing collusion attacks, which occur when several malicious cloud-based IDSs collaborate to give misleading judgments. Details are given in Chapter 5.

3.2.2.2 Evaluation

The proposed model was implemented using Matlab. We compared the proposed aggregation approach with other known aggregation approaches in the state-of-the-art which are : majority aggregation model [62] and weighted average aggregation model [96]. In the majority-based model, the IDS collects feedback from IDSs about suspicious behaviour and the decision is made (i.e., attack or not) according to the majority. However, in the weighted average aggregation model, weights W are assigned to feedbacks from different IDSs to distinguish their detection capability. Highly trusted IDSs are assigned with larger weights compared to low trusted IDSs. The results show that the proposed aggregation model enhances the accuracy, false positive, and false negative compared to the majority and weighted aggregation models. Details about the experiments and results are given in Chapter 5.

3.2.3 Formation of trustworthy federated clouds

We used the concept of cloud federation to enable trust-based coalition at the level of CPs. Thus, a CP can outsource some of its workloads to other CPs. This serves us in two purposes. First, reducing the extra overhead during the monitoring and detection process. Secondly, exploiting the power of other cloud-based IDS in handling sophisticated and severe attacks

since other clouds may have better investment (in terms of hardware and software security) in their intrusion detection solutions.

3.2.3.1 Proposed solution

We designed a trust-based framework for Cloud Federation (CF) formation. Our model enables a CP to evaluate other CPs trustworthiness by considering two approaches : objective and subjective trust evaluations. In the former, Bayesian inference was used to compute trust values based on previous interactions. In the latter, the DST integrated with the Bayesian inference was used to compute trust values in the absence of previous interactions. Thereafter, a novel decentralized algorithm was devised, based on a coalitional game theory, that allows heterogeneous CPs to establish their coalitions in such a way that maximises the trust of the formed federations. The proposed algorithm converges to a Nash-stable situation. More details are given in Chapter 6.

3.2.3.2 Evaluation

To evaluate the performance of the proposed approach, we tested the ability of the proposed method to enhances CPs performance in terms of availability, response time and throughput in the presence of untrusted (malicious or not) CPs. We used Cloudsim [97], based on the java programming language, for implementing our model.

The results show that the proposed trust-based federation model enhances throughout, availability and response time compared to the state-of-the-art QoS-based federation [82] and Grand Federation [74]. The results also show that the proposed model reduces the chances of untrusted (malicious or not) members in the federation and minimizes the chances of cooperating with unnecessary CPs. This, in turn, minimizes the resource cost compared to Grand and QoS-based federations. More details about the experiments and results are given in Chapter 6.

3.2.4 Fairness-assurance in multi-cloud cooperative IDSs

We proposed a fairness assurance mechanism in order to reduce the existence of selfish cloud-IDSs in the formed communities. This is useful to encourage cloud-based IDSs to participate in the community and reduce unnecessary consultation requests that can be exploited to exhaust well-behaving cloud-based IDS resources.

3.2.4.1 Proposed solution

The proposed fairness assurance mechanism is modeled as a Stackelberg game [17] in which each well-behaving IDS plays as the leader of the game and the selfish IDSs act as followers. The strategy of the selfish IDSs is to maximise their consultation rates and at the same time minimise their response rates. Knowing this strategy, the strategy of the well-behaving IDSs is to choose the optimal response rates that are fairly compatible with their consultation rates. We solved the optimization problem using the backward induction reasoning [98], through binding at the beginning the best response of the selfish IDS to the well-behaving IDSs consultation rate strategy, and then merging this information into the well-behaving IDSs optimization problem. The outcome of the game is the optimal response rate for the well-behaving IDS. More details are given in Chapter 7.

3.2.4.2 Evaluation

To evaluate the proposed fairness assurance mechanism, we made some IDSs to behave selfishly by making them not answer other IDSs consultation requests, while at the same time keeping on sending consultation requests.

The results show that the proposed model reduces the chances of cooperating with selfish IDSs. Moreover, the results show how the proposed fairness assurance mechanism can be used to minimize the selfish behavior. Selfish IDSs have no incentive to behave selfishly by maximising their consultation rates and at the same time minimising their response rates. More details about the experiments and results are given in Chapter 7.

3.3 Phase 3 : Proactive multi-cloud cooperative IDSs

In the third phase, our efforts were directed towards the achievement of objectives 7 and 8. We proposed a proactive multi-cloud cooperative IDS. The proposed model allows us to exploit the historical feedback received to produce learned models used for predicting the status (attack or not) of suspicious intrusions, even in the case of missing feedback. The above mentioned objectives were attained in the article titled : “A Deep Learning Approach for Proactive Multi-Cloud Cooperative Intrusion Detection System”

3.3.0.1 Proposed solution

The proposed solution is based on the Stacked Denoising Autoencoders (SDAE) approach, where a denoinig autoencoder is used as a building block to train a deep network [18][19].

Our model exploits the fact that a denoising autoencoder can learn how to reconstruct the original inputs, given partial data inputs, by allowing deep neural networks to learn (during the unsupervised pre-training stage) how to extract features that are robust to incomplete IDSs feedback. Such robust features can be seen as useful representations of data to yield a better intrusion detection accuracy in such real-time environments. This makes our detection model to be proactive at two levels : (1 by making decisions about suspicious intrusions even with missing feedback, and also (2 by making decisions without the need to apply any aggregation method on consulted IDSs feedback. It is worth mentioned here that the proposed model reduces the overhead associated with the architecture of multi-cloud cooperative IDS. The overhead is mainly due to the time taken by monitoring and analysing IDSs' incoming feedback. In traditional approaches, an IDS waits until receiving the whole feedback, then aggregates them in order to make a final decision about a suspicious intrusion. As a result, the overhead will continue to increase as long as the whole feedbacks have not yet received. We address this point by enabling an IDS to proactively make decisions about suspicious intrusions, even in the absence of complete feedback from consulted IDSs. This reduces the time of monitor and analysis of IDSs' feedback (i.e., overhead). More details are given in Chapter 8.

3.3.0.2 Evaluation

Our proposed proactive approach was implemented using TensorFlow. The results show the effectiveness of the proposed proactive model in making decisions in the presence of incomplete feedback. The average accuracy of the proposed model, at different numbers of hidden units, was slightly degraded. This indicates that the proposed machine leaning-based approach can effectively make correct decisions about suspicious intrusions, even in the absence of complete feedback from consulted IDSs. The results also show that the proposed model enhances the accuracy of the detection under partial inputs, compared to other state-of-the-art deep learning approaches. Details about the complete experiments and results are given in Chapter 8.

.

CHAPTER 4 ARTICLE 1 : AN SVM-BASED FRAMEWORK FOR DETECTING DOS ATTACKS IN VIRTUALIZED CLOUDS UNDER CHANGING ENVIRONMENT

Adel Abusitta, Martine Bellaïche and Michel Dagenais
Journal of Cloud Computing, vol. 33, pp. 55-65, 2017.

Abstract

Cloud Computing enables providers to rent out space on their virtual and physical infrastructures. Denial of Service (DoS) attacks threaten the ability of the cloud to respond to clients requests, which results in considerable economic losses. The existing detection approaches are still not mature enough to satisfy a cloud-based detection systems requirements since they overlook the changing/dynamic environment, that characterises the cloud as a result of its inherent characteristics. Indeed, the patterns extracted and used by the existing detection models to identify attacks, are limited to the current VMs infrastructure but do not necessarily hold after performing new adjustments according to the pay-as-you-go business model. Therefore, the accuracy of detection will be negatively affected. Motivated by this fact, we present a new approach for detecting DoS attacks in a virtualized cloud under changing environment. The proposed model enables monitoring and quantifying the effect of resources adjustments on the collected data. This helps filter out the effect of adjustments from the collected data and thus enhance the detection accuracy in dynamic environments. Our solution correlates as well VMs application metrics with the actual resources load, which enables the hypervisor to distinguish between benignant high load and DoS attacks. It helps also the hypervisor identify the compromised VMs that try to needlessly consume more resources. Experimental results show that our model is able to enhance the detection accuracy under changing environments.

4.1 Introduction

Several major Information and Communications Technology (ICT) companies are competing for creating advanced cloud computing services that are able to deal with small, medium-sized and large-scale enterprise demands. Many companies, organizations and governments are expected to transfer, if not already done, all or parts of their IT solutions to the cloud [4] [99]. This transfer is profitable from an economic point of view since it allows them to streamline

the spending on technology infrastructure and capital cost. However, the security threat in terms of Denial of Service (DoS) attacks constitutes a major obstacle against the achievement of this transfer. A DoS attack can be of many types and may be seen in different contexts (e.g., application, web services, network) [1]. However, in this paper, we consider Virtual Machine (VM)-based DoS attacks in a virtualized cloud and define a DoS attack as follows. A DoS attack occurs when one or more VMs drain all the available physical resources such that the hypervisor would not be able to support more VMs [3]. This attack is mainly caused by virtualization [2] [3], which is the backbone of the recent cloud computing architecture, where virtualization allows emulating a particular computer system and sharing physical resources (e.g., CPU and network bandwidth). In this paper, we shed light on the problem of detecting cloud-based DoS attacks under a changing environment. Although several advanced approaches have been proposed to detect DoS attacks in virtualized cloud (e.g., [27] [33] [34] [35]), these approaches still causes a significant decrease in the detection accuracy when used in a cloud environment. The reason is that the current approaches do not consider the changing environment, that characterises the cloud as a result of its inherent characteristics (resources restriction and scaling). Such characteristics are essential for the VM to meet the requirements of the pay-as-you-go business model [4].

4.1.1 Motivating Example

Assume that a cloud provider trained an Support Vector Machine (SVM) classifier on some of the features of the VMs under a certain infrastructure. These features include CPU, network, memory and I/O load. Assume now that the cloud provider, due to some business factors, decides to adjust some of the resources of the VMs. This adjustment includes revoking 45% from some of the resources of the VMs. Such an adjustment will result in a significant decrease in the DoS detection accuracy rate. The reason is that the features used to train the SVM classifier were extracted under the original infrastructure (before revoking 45% from VMs resources). However, these features become unsuitable in the light of the new adjustment in the VMs resources. In other words, the collected data will be affected by the new adjustment, which will lead to an inaccurate classification of the collected data. Tables 4.1 and 4.2 show our results of testing the impact of applying resources adjustments on the basic resources of the VMs (CPU, Memory, I/O and Network). We used the API of libvirt that employs cgroups [100] to adjust and limit the resources of the VMs. Using cgroups allows us to exploit Linux Kernel features which limit and allocate resources to VMs—such as CPU time, system memory, network bandwidth, or combinations of these resources [101]. The results show that the detection rate has been decreased as a result of revoking/granting resources from/to the VMs. The details of this experiment are described in Section 4.5.

Indeed, the continuous requests to make adjustments on the infrastructure are necessary as long as the cloud client (e.g., VM) wants to meet the Quality of Service (QoS) requirements. The reason is that the ability of performing new adjustments, to cope with the real-time economic factors, affects the decision of the industries, organizations and governments on whether to adopt or not cloud computing. In other words, the continuous adjustments are necessary for the continuous use of the cloud to meet the variations in the demands and the cost-efficiency, which are considered as the main cloud features.

Table 4.1 Attack detection rates when revoking resources (CPU, Memory, I/O and Network) from VMs.

Resources revoked from VMs	**Attack detection rate**
0% (baseline)	95.02%
10 %	95.79 %
20 %	90.28 %
40 %	89.08 %
60 %	85.18 %
80 %	83.67 %

Table 4.2 Attack detection rates when granting resources (CPU, Memory, I/O and Network) to VMs.

Resources granted to VMs	**Attack detection rate**
0% (baseline)	95.02%
10 %	95.79 %
20 %	85.28 %
40 %	84.08 %
60 %	75.18 %
80 %	74.67 %

4.1.2 Our Proposed Solution

To address the aforementioned problems, we propose a flexible detection framework based on the SVM learning technique. SVM is a classification technique that employs a nonlinear mapping to convert the original data into higher-dimensional of data, in order to find a hyperplane that optimally separates the training tuples based on their classes [46]. Our framework can be summarized as follows. The hypervisor collects some features to train the SVM classifier to be able to distinguish between the normal activity and DoS attack on the VM. The hypervisor then monitors and quantifies the effect of performing resources adjustments (i.e., granting/revoking resources to/from the VMs) on the collected VMs performance data. This information (i.e, effect of performing resources adjustments) is used thereafter to maintain

a filter of resources adjustments effect. The filter is used as a preprocessing step, prior to classification, to get rid of the "noise" that may show up on the collected data (due to the new adjustments) and that may considerably decrease the accuracy of the detection.

Moreover, the proposed framework enables VMs to regularly declare their current application metrics, such as number of clients, requests and sales. This is then used by the hypervisor to correlate these metrics with the actual resources load. This correlation enables the hypervisor to distinguish between benignant high load and DoS attacks. In addition, it enables the hypervisor to identify the compromised VMs that may try to claim and consume more resources. We propose a correlation technique that the hypervisor uses to calculate the expected resources load of the current compromised VMs based on the declared metrics. The calculated resources load is then compared with the actual resources load. If the calculated resources load is not within a certain range of the actual resources load, the belief that the VM has been compromised increases. In summary, we propose a comprehensive framework that consists of the following contributions :

— Proposing a detection approach to identify DoS attacks in a virtualized cloud under changing environment. To the best of our knowledge, our work is unique in considering the detection problem under changing environment in virtualized clouds.
— Proposing the monitoring and quantification of the effect of performing resources adjustments, which enhances the accuracy of identifying DoS attacks under changing environments.
— Proposing a model to correlate VMs metrics with the actual resources load by the host, which enables the hypervisor to identify compromised VMs.
— Modeling an incentive technique that enables the hypervisor to give incentives in the form of resources to the VMs that have truthfully declared their metrics and punish these VMs that lied about their actual metrics.

4.1.3 Paper Outline

The rest of the paper is organized as follows. In Section 4.2, we discuss the related work. In Section 4.3, we present the proposed framework. Section 4.4 presents security analysis of the proposed framework. In Section 4.5, we present our empirical results. Finally, Section 4.6 concludes the paper.

4.2 Related Work

Machine learning for detecting DoS attacks in the cloud was used by several researchers. This work benefits from many advanced machine learning and and artificial intelligence techniques to predict the status of VMs (i.e, malicious or normal). Lonea et al. [24] uses the normal traffic pattern received from the Virtual Bridge (VB) of the VM to validate for consistency against behavioral patterns of attacks. They use a network intrusion detection system, that analyzes the normal traffic flow obtained from the VB, to check and test for consistency against the attack behavioral patterns. Thus, if abnormal traffic has been detected, the anomaly information will be reported and an alarm generated.

Similarly, Gupta et al. [27] propose a profile based network intrusion detection system. They combine both fine grained data analysis and Bayesian techniques in order to detect TCP SYN flooding. The main advantage of these approaches is the ability to identify DoS symptoms at an early stage, because their approaches are able to collect information at the networking level, before the DoS causes a significant performance degradation. However, the lack of application performance information can result in wrongly identifying high traffic during the peak time as a DoS attack.

Ficco et al. [32] propose a strategy for generating stealthy DoS attacks in the cloud, which uses low overhead attacks to inflict the maximum financial cost to the cloud clients [32]. Masood et al. [33] propose a web-behavior-based detection, where they identify two client's profiles. The first one is for good clients while the second one is for bad clients. A good client will follow a pattern that reflects normal activity on the web, while a bad client will show some abnormal activities. Similarly, Anusha et al. [34] study the behavior of normal users of Web applications. They assume that an attacker spends a very short time (almost zero) over a Web page. They use for that a metric called Time Spent on a Page (TSP). They assume that the attackers TSP is very close to zero. In contrast, the TSP of a normal client should be high enough to interact with the Web page. The work of Kwon et al. [35] also uses a behavioral approach for detection. They start from a tested assumption saying that the behavioral patterns of normal traffic are similar, while the behavior patterns of malicious traffic are not. The cosine similarity is used to check the similarity of the traffic. If such a similarity does not exist, an alarm is generated. A main advantage of this approach that it is able to determine the similarity during run-time. However, there is no guarantee that the normal traffic will always be similar in a dynamic environment (i.e. cloud). In fact, in some applications, we could have many forms of normal traffic that are fairly dissimilar.

Palmieri et al. [36] use a two-phase ML-based detection technique. The first phase is called

Blind Source Separation (BSS), while the second phase is called Rule-based Classifier to detect zero-day attacks that change or alter the traffic volume rate. BSS extracts the features of the cloud nodes traffic in order to be used by a decision tree classifier to create a normal traffic profile (baseline). Most recently, Choi et al. [37] propose a data-mining-based approach to detect application layer HTTP GET DoS attacks. They use a normal behavior pattern to detect DoS attacks on VMs. The parameters used for analyzing and creating attack patterns are : CPU usage, packet size and packet header information. They evaluate their approach by comparing it with a signature-based approach. The result showed that their proposed method performs better than SNORT in terms of identifying new attack profiles. Similar to this work, Jeyanthi and Mogankumar [38] and Jeyanthi et al. [39] propose a mechanism to detect DDoS attacks based on clients request rate. Clients requests will be put in a black list or white list based on a certain threshold rate. The threshold is determined by calculating the maximum number of legitimate client requests. The authors have shown experimentally that the legitimate clients could continue being served during an attack using their method. However, the main disadvantage of this method is that it is threshold based, where setting the optimal threshold is always difficult and infeasible in a production cloud environment.

Chonka and Abawajy [40] and Chonka et al. [41] propose a decision tree classification technique. The method operates in two phases : training phase (first phase) and testing phase (second phase). In the training phase, a rule set that has been generated over time by the decision tree classifier is used to define both known and unknown attributes. In the testing phase, a decision making module is used to decide the likelihood of a previously classified packet. This helps decide whether to let a packet enter or not. Similar to that, Lonea et al. [24] also proposed a classification technique based on Intrusion detection system (IDS). The detection module analyse the alerts generated by each VM using the Dempster-Shaferther theory (quantitative solution classifier) in 3-valued logic and fault tree analysis (FTA). Although Dempster-Shaferther is able to produce powerful results when observations about attacks come from different sources, it becomes unsuitable when one source produces multiple observations [42].

Among other approaches, the work of Iyengare et al. [43] proposes a Multilevel Thrust Filtration (MTF) that contains four detection and prevention modules, which are traffic analysis, anomaly detection, anomaly classification, and attack prevention. The proposed method filters the incoming packets and detects four types of traffic congestion, which are spoofing, ash crowd, DDoS, and aggressive legitimate traffic. A similar approach has been proposed by Jeyanthi and Iyengar [39]. The main feature of this approach is the ability to increase the attack detection accuracy because multiple stages of detection are used. However, it is associated with a significant overhead since multiple algorithms and techniques should be

used.

Cooperative IDSs in cloud have been proposed in several works. For example, Teng et al. [61] propose an approach that aggregates two different detectors : feature and statistical detectors. The feature detector adopts SNORT to separate events based on Transmission Control Protocol (TCP). The statistical detector cooperates with SNORT by using data packets from it to find whether an event is an attack or not. If the rate of packets obtained exceeds the predefined threshold, this means that there is an actual attack. Similarly, Man and Huh [63] and Singh et al. [64] propose a cooperative IDS between different cloud regions. Their approach enables exchanging alerts from multiple places (detectors). The proposed approach allows the exchange of security information between interconnected clouds. Ghribi [65] proposes a middleware IDS. The approach allows a cooperation between three layers : Hypervisor-based, Network-based, and VM-based IDS. If an attack was found in a layer, it cannot be executed in the other layers. Chiba et al. [66] also propose a network-based cooperative IDS to identify network attacks in the cloud environment, which is performed by monitoring traffic while maintaining performance and service quality. Recently, Wahab et al. [102] [103] propose a game theoretic-based IDS. The approach that they used enables a CP to optimally distribute its resources among VMs in such away to maximize the detection of distributed attacks. The main limitation of the cooperative IDS is that they work in the assumption that all nodes are trustable, which makes it vulnerable to malicious insiders.

Cloud-based DoS attacks mitigation approaches are also proposed in several works. For example, Yu et al. [104] propose a dynamic resource allocation feature for VMs. This allows attacked VMs to acquire extra resources during DoS attacks. When a DDoS attack occurs, the cloud hires the idle resources to clone sufficient attack prevention servers for the attacked VMs in order to guarantee the quality of service for the users by filtering out attack packets. This approach is beneficial in an environment where DoS attacks are frequently generated. However, it can be exploited by selfish VMs to acquire and use much resources even though there is no attack. Also, Somani et al. [105] propose auto-scaling decisions by differentiating between legitimate and attack traffic. Attack traffic is detected based on the workload of human behavior. The advantage of this approach is that it gives more attention to serve legitimate clients by making accurate and proper autoscaling decisions. However, a workload of human behavior can be emulated. This makes an attacker able to deplete cloud resources.

The summary of the existing works is given in Table 4.3. The aforementioned works paved the way to understand the issues behind improving the detection accuracy in cloud environments. Our proposed model offers two major features. The first is the aspect of detection under

changing environment. To the best of our knowledge, our work is the first to consider the detection problem under changing environment in virtualized clouds. The second is allowing VMs to share information about their current application metrics (e.g, number of clients, requests and sales) to the hypervisor, which in turn allows distinguishing between legitimate high load and DoS attacks. It also enables the hypervisor to identify the compromised VMs that try to claim and consume more resources.

Table 4.3 Cloud-based Detection Approaches.

Researchers	**Approach**
Lonea et al. [24]	Profile-based detection
Gupta et al. [27]	Profile-based network detection
Masood et al. [33]	Web-behavior-based detection
Anusha et al. [34]	Web-behavior-based detection
Kwon et al. [35]	Web-behavior-based detection
Palmieri et al. [36]	Machine learning-based detection
Choi et al. [37]	Data Mining-based detection
Jeyanthi and Mogankumar [38]	Data Mining-based detection
Jeyanthi et al. [39]	Data Mining-based detection
Iyengare et al. [43]	Multiple Stage detection
Jeyanthi and Iyengar [39]	Multiple Stage detection
Teng et al. [61]	Cooperative-based detection
Man and Huh [63]	Cooperative-based detection
Singh et al. [64]	Cooperative-based detection
Chribi et al. [65]	Cooperative-based detection
Wahab et al. [102] [103]	Game theocratic-based detection

4.3 The Proposed Framework

In this section, we describe the key constituents of our framework. The framework contains a set of components, each of which exhibits a set of modules. Figure 4.1 illustrates the framework structure and describes the modules of each component. These components include : data gathering, data and load analysis, and detection.

We use the Linux Trace Toolkit next generation (LTTng) [92] to gather the VMs performance metrics. LTTng is a powerful, low impact and lightweight [93] open source Linux tracing tool. It provides precise and detailed information on the underlying kernel and user-space executions. LTTng contains different trace points in various modules of the operating system kernel. Once a predefined trace point is reached, it generates an event containing a time-stamp, CPU number and other run-time information related to the running processes.

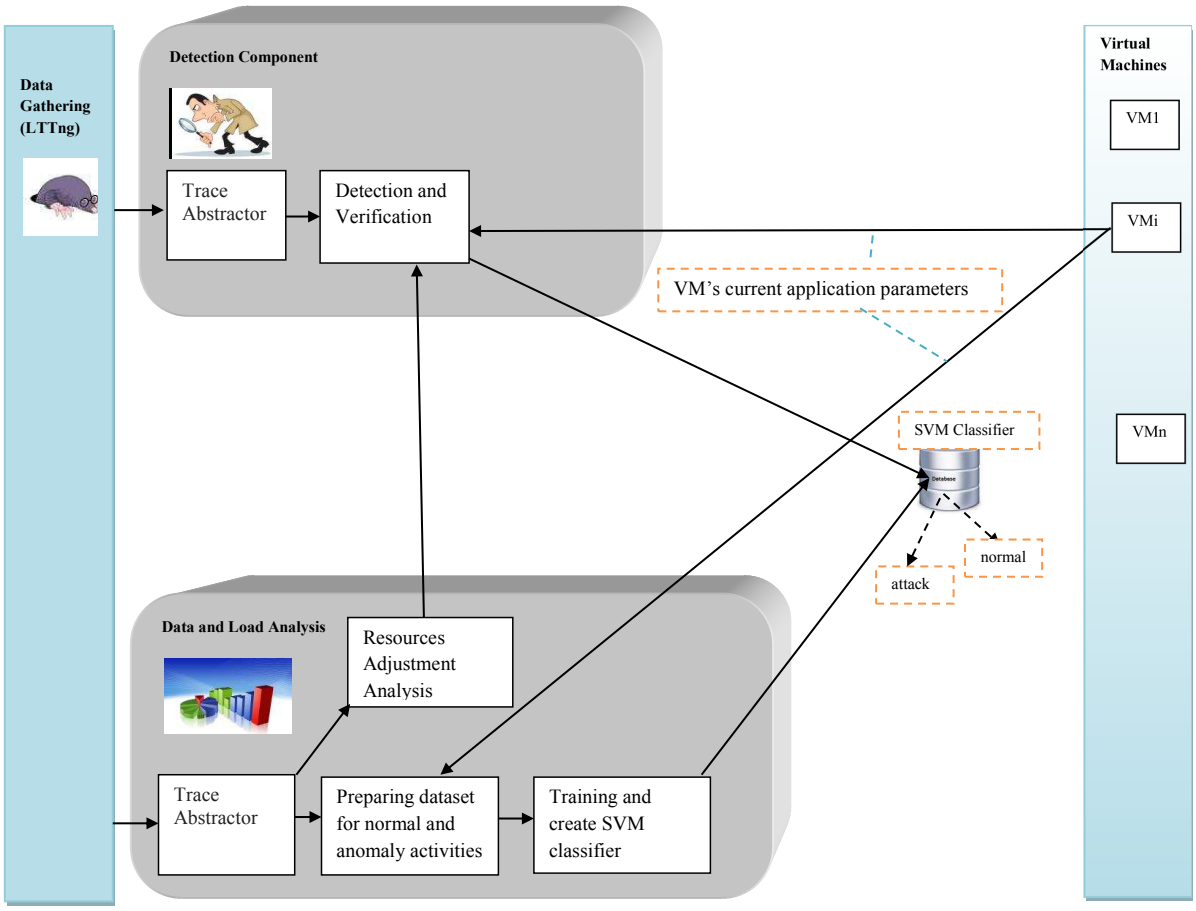


Figure 4.1 Architecture of the Proposed Framework

4.3.1 Data Analysis

This component is responsible for analysing data obtained from the data gathering component. We divide this component into the four modules : Trace abstractor, preparing dataset for normal and anomaly activities, training and create SVM classifier, and resources adjustment analysis.

4.3.1.1 Trace Abstractor

The trace file size is usually so large that it is difficult to analyze and understand the system execution. Most of the time, another analysis tool is required to abstract the low-level events and represent them as higher-level events, thus reducing the data to be analyzed. Trace abstraction is typically required to compute statistics of complex system metrics that are not directly computable from the low-level trace events [106]. For instance, to compute synthetic metrics such as ”number of HTTP connections”, ”CPU usage”, and ”number of different

types of system and network attacks", raw events must be aggregated to generate high-level events. Then, the desired metrics must be extracted and computed. Table 4.4 gives examples of a higher-level event generated from low-level events. The details of the trace abstraction tool used to generate such high-level meaningful events can be found in [93].

Table 4.4 Abstracting Example.

Low-level events	**higher-level event**
socket create	HTTP connection
socket call	
socket send	
socket receive	
socket close	

4.3.1.2 Preprocessing and Training Modules

In this phase, the SVM [107] classification technique is used to analyse the collected data and classify the VMs load. SVM is a classification technique that employs a nonlinear mapping to convert the original data into higher-dimensional data in order to find a hyperplane that best separates the training tuples based on their classes. The hyperplane is determined using support vectors and margins in such a way that maximizes the hyperplane's margins with the aim of delivering more accurate results when classifying future data tuples [46]. We use SVM thanks to its ability to generate very accurate classifiers (especially in binary classifications) [107] and its effectiveness in high dimensional datasets consisting of a large number of attributes [108]. Moreover, it is robust against outliers and overfitting [109] [110].

The training dataset is generated during the monitoring of the host to determine which metrics reflect the malicious behavior and which ones reflect the normal behavior. As shown in the proposed architecture (Figure 4.1), VMs are allowed to share their application/business metrics to be considered among the features used to train the SVM. The VM application/business metrics are discussed later in this paper.

Each VM can be either under DoS attack or normal. Thus, class label $y_i \in \{\text{attack, normal}\}$. Given the training datasets $(x_i,y_i)...(x_n,y_n)$, x_i is the VM metrics values used for the training. n is the number of metrics values, the objective is to find the hyperline that offers a maximum margin (Figure 4.2) such that :

$$w * x + b = 0 \tag{4.1}$$

Where w is a weight vector and b is a threshold.

Thus, the training data should satisfy :

$$w * x_i + b \geq -1 \quad for \quad all \quad attack \quad data \quad x_i \tag{4.2}$$

$$w * x_i + b \leq +1 \quad for \quad all \quad normal \quad data \quad x_i \tag{4.3}$$

The problem is converted to finding the optimal hyperplane (Eq. 4.1), which can be turned into a convex optimization problem [46] :

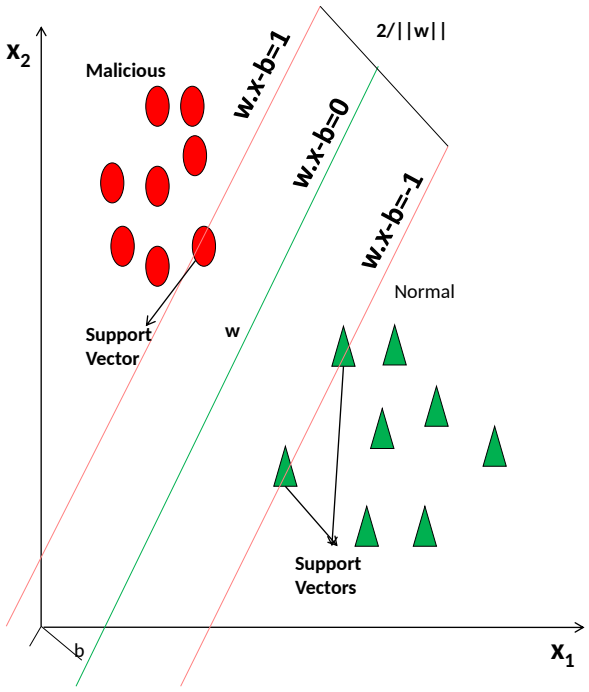


Figure 4.2 The value of w affects the position of the hyperplane.

$$\left\{ \begin{array}{l} min \quad \tau(w) = \dfrac{\|w\|}{2} \\ Subject \quad to \quad y_i(\langle w, x_i \rangle + b) \quad for \quad all \quad i = 1, ..., n \end{array} \right\} \tag{4.4}$$

The convex optimization can be solved using Lagrange multipliers [46] :

$$\left\{ \begin{array}{l} maximize \quad L(\alpha) = \displaystyle\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ \\ Subject \quad to \quad \displaystyle\sum_{i=1}^{n} y_i \alpha_i = 0 \quad and \quad 0 \leq \alpha_i \leq C \quad \forall \quad 1 \leq i \leq n \end{array} \right\} \tag{4.5}$$

where α_i is the Lagrange multipliers [46], $K(x_i,x_j)$ represents the kernel function (e.g., linear, polynomial, etc.) and C is a constant for determining the trade-off between margin maximization and training error minimization [46].

By solving Eq. 4.5 we get [46] :

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{4.6}$$

Finally, the decision attack function is given by :

$$f(x, \alpha, b) = \{\pm 1\} = sgn(\sum_{i=1}^{n} y_i \alpha_i K(x, x_i) + b) \tag{4.7}$$

4.3.1.3 Resources Adjustment Analysis

The impact of resources adjustment appears on the collected data, which makes the SVM classifier unsuitable in the light of the new adjustment in the VMs resources. In other words, the collected data will be affected by the new adjustments, which leads to an inaccurate classification of that data. To address this issue, we should determine the effect of resources adjustment on the collected data (we will discuss later in this section how to calculate the effect of resources adjustments). The effect of resources adjustment reflects to what extent the modified data (after new resources adjustment) deviates from the original data (the data that meets the basic infrastructure).

Having the effect of resources adjustment, two approaches to solve the detection problem under changing environment can be considered. The first approach is to consider this effect during the training of the SVM classifier. This can be done by generating new sub features for every feature (features represent system metrics in this context) used in the training. In fact, the new sub features are the result of applying the effect of resources adjustment on the basic feature. For example, if the original feature used to train the SVM classifier is 10, 20, 30, 40 for CPU, memory, I/O, and network, respectively, then the SVM classifier is also trained to classify in the presence of 50% adjustment on the VM resources by adding to the training set the following new sub feature : (10 + 10 * 50%), (20 + 20 * 50%), (30 + 30 * 50%) and (40 + 40 * 50%) for CPU, memory, I/O, and network respectively. This makes the SVM classifier be trained not only on the original infrastructure but also on the new infrastructure after resources adjustment.

The second approach is to account for the resources adjustments effect by a separate filter. The filter is used as a preprocessing step, prior to classification, to get rid of the effect of

resources adjustment on the collected data, in order to normalise the data with respect to the original infrastructure on which the training was performed, before passing it to the SVM classifier. We adopt the second approach in the proposed framework for the two following reasons. On the one hand, the first approach requires generating a huge training dataset, since we have to generate many sub-features for each single feature. This results in more overhead during the SVM training. The second approach does not require any change in the dataset. On the other hand, training an SVM with all possible adjustments (as is the case for the first approach) may lead to an overfitting. Specifically, the classifier might work correctly in the presence of the trained adjustment ; However, the classifier accuracy will go down in the absence of such adjustments.

4.3.1.4 The Effect of Resources Adjustment.

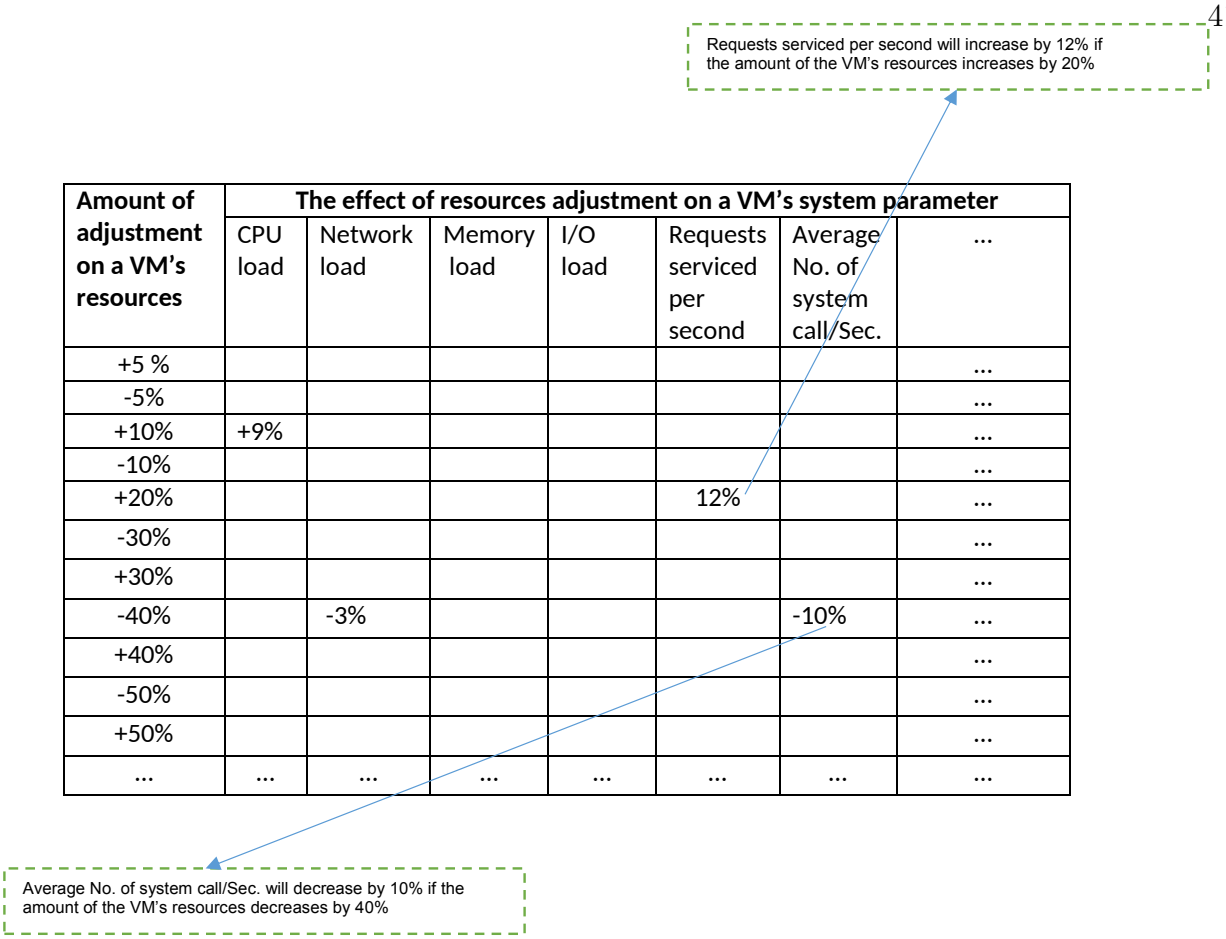
The effect of resources adjustment on VMs is studied during the running of the VMs to find out to what extent their system metrics are affected by the adjustments. Although such a process requires changing the resources of the VMs which in turn may affect the performance of the application running inside these VMs, the impact of an adjusting and monitoring the effect is acceptable since it is done during a short time period ($\approx$ the time needed to capture VMs system metrics). Note that resources adjustment process can be done by exploring and monitoring the effect of all possible resources adjustments performed on the system metrics of the VMs. More specifically, we maintain a filter of resources adjustments effect (as in Figure 4.3), which is used as a preprocessing step prior to classification (i.e., SVM), to filter out the effect of resources adjustments from the collected data. In other words, this helps to get rid of the noise that may show up on the collected data (due to the new adjustments) and may considerably decrease the accuracy of the detection. Algorithm 1 is used to determine the effect of all possible resources adjustments on the system metrics of the VMs.

In Algorithm 1, for each VM^j ($j \in$ VMs) in a certain host, the algorithm monitors and determines the current system metrics of j to be stored in array U_1 [] (line 3). Then, for each possible resources adjustment adj, the algorithm applies resources adjustment of value adj, to see the effect of adj on j's system metrics. The list of all possible resources adjustments, which is given as an input in Algorithm 1, can be selected manually by the cloud administrator. The administrator can consider these adjustments that have significant impacts on VMs' system metrics and also had been requested in the past by the client according to the pay-as-you go business model. This, in turn reduces unnecessary study of unneeded adjustments and thus decreases the processing time of Algorithm 1. Note that we apply adjustments on VMs resources using control groups (cgroups) [101], which is a Linux Kernel feature that limits

Algorithm 1: Determining the effect resources adjustments on VMs system metrics.

Input : List of VMs' possible resources adjustments $adj[]$
Output: The effect of resources adjustment on VMs system metrics
$\{eff_1[][], eff_2[][], \ldots, eff_n[][]\}$

```
repeat
    foreach Running VM j do
        Monitor and store j' system metrics in array U_1 (before adjustments)
        foreach Adjustment adj ∈ adj [] do
            Apply an adjustment of amount adj on j's resources
            Monitor and store j' system metrics in array U_2
            Remove the adjustment of amount adj and restore j's default resources
            foreach index i of U_1 do
                BeforeAdj = U_1[i]
                AfterAdj = U_2[i]
                if |BeforeAdj − AfterAdj| < ε then
                    eff_j[adj][i] = 0
                else if BeforeAdj < AfterAdj then
                    eff_j[adj][i] = (adj*100)/AfterAdj * BeforeAdj
                else
                    eff_j[adj][i] = - (adj*100)/AfterAdj * BeforeAdj
                end
            end
        end
    end
until ε elapses;
```



Amount of adjustment on a VM's resources	The effect of resources adjustment on a VM's system parameter						
	CPU load	Network load	Memory load	I/O load	Requests serviced per second	Average No. of system call/Sec.	...
+5 %							...
-5%							...
+10%	+9%						...
-10%							...
+20%					12%		...
-30%							...
+30%							...
-40%		-3%				-10%	...
+40%							...
-50%							...
+50%							...
...	...	...	...	...	...	...	...

Figure 4.3 Table used for filtering out the effect of resources adjustments on a VM system metrics.

and allocates resources to VMs — such as CPU time, system memory, network bandwidth, or combinations of these resources. The system metrics are re-computed after each adjustment process (line 6). Then, the algorithm computes the effect of an adjustment adj on VM j's system metrics by finding the percentage of change on the j's system metrics (line 8 - 16). If the new calculated metric U_2 $[i]$'s value is within a small range of the old value $U_1[i]$, the effect will be considered as 0. This is described in the Algorithm 1 as $eff_j[adj][i] = 0$ (line 12), which means that the effect of an adjustment adj on that i-th metric of VM j is equal to 0. This indicates that the resources adjustment had no effect on that given metric. However, if the new calculated metric's value considerably differs from the old one, the algorithm computes the percentage of change on the old value such that : $eff_j[adj][i] = \frac{adj*100}{AfterAdj} * BeforeAdj$ (line 14) for a positive change (i.e., new-value $>$ old-value) and - $\frac{adj*100}{AfterAdj} * BeforeAdj$ (line 16) for a negative change (i.e., new-value $<$ old-value). The algorithm is used for each possible adjustment in order to have a filter of resources adjustments effect for each VM. The filter is then used during the detection step to filter out the collected data from the effect of resources adjustment. This adapts the environment of the detection to the original environment (i.e.,

the environment in which the SVM classifier was created). Note that the whole process is repeated periodically after a certain fixed period of time ε (line 17) to capture the new effect of the given resources adjustment on VMs system metrics.

It should also be noticed that some resources adjustment may have no impact on the system metrics of the VMs. In fact, it can depend on resources and the underlying usage model. For example, for a given load, if we consume 60% of the CPU (100% available with no restriction), when a restriction of 70% is imposed, we will consume 60% / 70% = 85% of the available CPU. The number of CPU seconds will remain the same though. For this purpose, we have considered in the resource adjustment algorithm the following condition : If ($|BeforeAdj - AfterAdj| < \epsilon$) (line 11), which means that the collected data are the same before and after adjustments. The effect of resources adjustment in this case is equals to 0, as given in Algorithm 1 ($eff_j[adj][i] = 0$).

4.3.2 Detection Component

This component is used for identifying DoS attacks. It monitors the system and performs tracing abstraction, similar to the steps used above. Along with these steps, the module performs the detection algorithm described in Algorithm 2. The following section presents the details of this component.

4.3.2.1 VM's Declared Application/Businesss Metrics

As shown in the proposed architecture (Fig. 4.1), the proposed detection component enables VMs to declare their current application and/or business metrics, such as number of clients, requests and sales, to the hypervisor. These metrics are used during the generation of the training dataset due to train the SVM classifier. In addition, it can be used by the hypervisor to correlate these metrics with the actual resources' load (network, CPU, disk, I/O) and decide whether it is coherent or not (compromised VM trying to needlessly claim and consume more resources).

We use the concept of simple linear regression to model the correlation between the VM's declared metrics and the actual resources' load. A simple linear regression can be used for better and more efficient fitting in modeling the relationship between the application metrics and the resources' load in cloud-based applications [111] [112] [113].

Suppose that there are n information about VM's declared application metrics and its corresponding resources' load $\{(VMT, load), i = 1, ..., n\}$, where VMT represents the former and $load$ represents the later. The function that describes VMT and $load$ is :

$$load_i = \alpha + \beta * VMT_i + \varepsilon_i \tag{4.8}$$

Our goal is to find the equation of the straight line such that :

$$load = \alpha + \beta * VMT \tag{4.9}$$

We calculate β and α using the least-squares approach by looking at a line that minimizes the sum of squared residuals ε of the linear regression model such that :

$$min \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (load_i - \alpha - \beta * VMT_i)^2 \tag{4.10}$$

The objective can be solved using inner product spaces [114] to find the value of α and β.

$$\beta = \frac{\sum_{i=1}^{n} (load_i - \overline{load})(VMT_i - \overline{VMT})}{\sum_{i=1}^{n} (load_i - \overline{load})^2} \tag{4.11}$$

$$\alpha = \overline{VMT} - \beta * \overline{load} \tag{4.12}$$

Where $\overline{VMT}$ is the mean of the dataset's of the VM's declared metrics and $\overline{load}$ is the mean of the resources' load.

When the hypervisor wants to decide whether there is a coherence between the declared metrics and the actual resources' load, it uses Eq. (4.9) by substituting α and β with Eq. (4.11) and Eq. (4.12), respectively. The calculated resources' load is compared with the actual resources' load to find if the calculated resources' load is within a small distance of the actual load or not.

Using such a model has two important advantages for the hypervisor. First, it allows the hypervisor to distinguish between high benignant load and DoS attacks. When the hypervisor receives metrics from the VM about receiving unusual high load, the hypervisor can calculate the resources' load based on the VM's high load declared metrics using Eq. (4.9). If the calculated resources' load is close to the actual resources' load, the hypervisor identifies the high load as a peak load. The second advantage of the model is that it allows us to identify the compromised-VMs that try to pretend receiving a large number of requests to be allowed to claim and consume more resources. This can be done also by calculating the resources' load of the current declared metrics using the model and compare it within the actual resources' load. If the VM has been compromised, the calculated resources' load, in most cases, will

be different from the actual resources' load as long as the compromised-VM has no detailed information about the model used to calculate the resources' load.

4.3.2.2 Detection Algorithm

Algorithm 2 that is used for detecting DoS attacks works as follows. For each VM^j ($j \in$ VMs) in a certain host, the algorithm uses the model obtained in the previous section to calculate the VMs resources load (calculated load) with respect to the declared metrics of the VMs (line 4). This step is important to minimise unnecessary false positive alarms during flash events. The calculated resources load is compared with the actual resources load to determine if the calculated resources load is within a small range of the actual resources load (line 5). If this is not the case, the detection process starts by filtering out the effect of resources adjustment on j's system metrics using the resources adjustment effect of j (given as input in Algorithm 2) (i.e., $newU[i] = U[i] \pm (U[i] * eff_j[VMA^j][i])$ (line 6-10) from the collected data. The objective is to adjust the data for the original infrastructure on which the training was performed before passing it to the SVM classifier. The Algorithm passes then the modified collected data to SVM to predict the result (line 11). If the result $r =$ "attack", the algorithm identifies a DoS attack (line 12-13). If the calculated resources load is within a short distance of the actual load, the algorithm identifies the resources load as normal (line 16-17).

The main complexity of the proposed approach lies in the SVM training, which is commonly known to be $O(n^3)$ [110], where n represents the training set size. Although this might be infeasible for very large datasets, the training process is performed only once, and its overhead can then be neglected [115] [116]. Moreover, recently, more and more techniques are being proposed for efficient SVM training [117] [118] [119]. As for the prediction process in Algorithm 2, the complexity lies in three parts. In the first part, the algorithm filters out the effect of resources adjustment from the collected data (line 6-10). The computation complexity for this part is $O(n)$, where n is the number of the input metrics of SVM. The second part is to find if the predicted and calculated VM resources load are similar (line 5), which is of constant time complexity $O(1)$. The last part of the algorithm is to predict the modified collected data using SVM (line 11). The computational complexity of SVM-based prediction is $O(n)$, where n is also the number of the input metrics of SVM. Therefore, the overall computation complexity of the proposed detection Algorithm is $O(n) + O(n) + O(1) \approx O(n)$.

Algorithm 2: Detection Algorithm

Input : VMs' alpha values $\{\alpha_1, \alpha_2, ..., \alpha_n\}$
Input : VMs' beta values $\{\beta_1, \beta_2, ..., \beta_n\}$
Input : VMs' declared application metrics $VMT^j \quad \forall\, j \in VMs$
Input : VMs' effect of resources adjustment on their system metrics $\{eff_1[][], eff_2[][], ..., eff_n[][]\}$
Input :Amount of adjustment applied to VMs resources $VMA^j \quad \forall\, j \in VMs$
Output :Attack Boolean Dec

foreach *VM j* **do**
  Determine j's current resources load crt_load.
  Monitor and store j's system metrics in array U.
  $calc\_load = \alpha_j + \beta_j * VMT^j$.
  if $|calc\_load - crt\_load| > \epsilon$ **then**
    foreach *index i of U* **do**
      if $eff_j[VMA^j][i] \geq 0$ **then**
        $newU[i] = U[i]$ - $(U[i] * eff_j[VMA^j][i])$
      else
        $newU[i] = U[i] + (U[i] * eff_j[VMA^j][i])$
      end
    end
    `/* classifying data (i.e., newU using the SVM */`
    $r = predict(newU, SVM)$
    if r==*"attack"* **then**
      Dec = True
    end
    else
      Dec = False
    end
  end
end
else
  Dec = False
end

4.3.2.3 Verification and Resources Allocation

The proposed detection framework allows a VM to regularly declare its current application metrics, which enables the hypervisor to correlate these metrics with the actual resources load and decide if it is coherent or not (compromised VM trying to claim and consume more resources). In fact, the VM may have no incentive to declare its metrics. Moreover, the VM may lie about its current metrics either because of its selfish strategy (in order to obtain more resources) or because the VM has been compromised.

Algorithm 3: Verification and Resources Allocation Algorithm

Initialisation:
Input : VMs' alpha values $\{\alpha_1, \alpha_2, ..., \alpha_n\}$
Input : VMs' beta values $\{\beta_1, \beta_2, ..., \beta_n\}$
Input : VMs' declared application metrics $VMT^j \quad \forall\, j \in VMs$
Output: Amount of resources granted/revoked to/from each VM

```
1  foreach VM j do
2  |  Determine j's current resources load crt_load
3  |  calc_load= α_j + β_j * VMT^j
4  |  if |calc_load − crt_load| < ε then
5  |  |  Grant resources to j
6  |  end
7  |  else
8  |  |  Revoke resources from j
9  |  end
10 end
```

To address the aforementioned problems, we propose a verification algorithm (Algorithm 3). Our solution motivates the VM to declare its current application metrics by granting resources to the VM whose calculated resources load (obtained from the VM's declared metrics) falls within a close range of the VM's actual resources load (line 4-5). On the other hand, the hypervisor revokes resources from the VM whose calculated resources load and actual resources load do not match (line 6-7). This dissimilarity, in most cases, is either because the VM has lied about its declared metrics, or because the VM has been compromised. The amount of resources revoked from the VM can be decided by the system administrator, who clearly knows the real impact of adjusting the VM's resources on their performance. However, we suggest that the amount be proportional to the magnitude of the difference between the calculated resources load and the current resources load. In other words, the larger the difference between the calculated resources load and current resources load is, the more resources should be revoked from the VM. This encourages VMs to truthfully declare their metrics used to calculate the resources load.

4.4 Security Analysis of the Proposed Framework

The main objective of the proposed framework is to enhance the detection of DoS attacks under changing environment. In this section, we analyse the effectiveness of our framework in the presence of flash events, DoS attacks and compromised VMs.

4.4.1 Flash Events

A flash event occurs when there is an unusual surge of legitimate traffic. Our model is able to distinguish between a flash event and DoS attacks since our framework allows VMs to declare their current application metrics (e.g., number of clients) and motivates them to do that by granting them extra resources (Algorithm 3). The declared metrics can represent flash events. The declared metrics are then used to calculate the resources load according to the model in Section 4.3.2.1. The calculated and actual resources load are then compared to see if they approximately match. If so, the hypervisor will know and understand that the VM is under an unusual surge of legitimate requests and will grant the VM more resources to serve better during this period. Otherwise, if the calculated and actual resources load are largely different, the hypervisor revokes some resources from the VM. This strategy motivates the VM to truthfully declare its peak load and limit the illegal use of resources by the attacker, in case the VM has been compromised.

4.4.2 DoS Attacks

The main purpose of the proposed framework is to detect DoS attacks. Our model achieves this by using SVM. The framework trains SVM classifiers on the normal and malicious features to achieve the learning of the classifier. Our framework then monitors the incoming features and predicts the system status. Moreover, the proposed detection algorithm supports the detection under a changing infrastructure. The algorithm checks if no modification in the VM's resources (i.e., granting or revoking resources) has occurred. If so, the algorithm passes the collected data directly to the SVM classifier, without applying any filtering strategy. Otherwise, the algorithm filters the effect of resources adjustments in order to adjust the collected data to the original infrastructure (on which the training was performed), before passing it to the SVM classifier. This is done by removing the effect of noise caused by resources adjustments.

Theorem 1. *The accuracy of the detection will not be affected after filtering out the effect of resources adjustment.*

Démonstration. Consider that the collected data U has been affected by resources adjustment in such a way that makes U change by percentage $\%eff$. The $\%eff$ can be a positive (e.g., +5%), negative (e.g., -5%) or 0 (as the example given in Section 4.3.1.3, where some adjustments (e.g., CPU) do not result in any change in the collected data). If $\%eff$ is positive, the value of U changes to $U + (\%eff * U)$. In this case, our detection algorithm removes the adjustment (Algorithm 2 line 5-9) as follows :

$$U + (\%eff * U) - (\%eff * U) = U \tag{4.13}$$

On the other hand, if $\%eff$ is negative, the value of U changes to U - (%eff*U). In this case, the detection algorithm removes the adjustment (As in Algorithm 2) as follows :

$$U - (\%eff * U) + (\%eff * U) = U \tag{4.14}$$

Also, if $\%eff = 0$, the value of U changes to U - (0 * U) = U. In this case, the detection algorithm removes the adjustment as follows :

$$U - (0 * U) + (0 * U) = U \tag{4.15}$$

In the aforementioned three situations, the collected data U can be recovered, which means that the detection will be performed as if no adjustment had been applied. □

4.4.3 Robustness against Compromised VMs

The proposed framework is able to detect the compromised VMs that try to claim receiving an unusual load of client requests, to be allowed to consume more resources. The compromised VM can hide that its compromised by mimicking normal load and/or flash crowds [120]. The hypervisor calculates the VM load (resources load) based on the compromised VM's current declared application metrics and compares it with the actual resources load. If the calculated resources load is not within a short range of the actual resources load, there will be a high probability that the VM has been compromised. A possible strategy that a compromised VM may use, is trying to find α and β in order to obtain the model for calculating the resources load $load = \alpha + \beta * VM\_par$. This can be done by using different values of α and β. For every α and β, the compromised VM sees the response from the hypervisor (the response is the resources given for the unusual declared metrics). If the compromised VM did not receive a response, the compromised VM tries other values of α and β, until the correct α

and β values are obtained (receiving a response from the hypervisor). Although this can be possible, the number of trials will be very high, which makes it infeasible for the attacker, since α and β can be any real number from a large interval. In addition, the attacker will typically not have the opportunity to do a large number of attempts in trying to guess α and β. After several wrong guesses (e.g., 10 wrong attempts), the hypervisor would consider the VM as a compromised and prevent it from using resources.

4.5 Experimental Results and Analysis

In this section, we first explain the experimental setup used to perform our experimentation and then study the performance of the proposed detection approach.

4.5.1 Experimental Setup

To evaluate our model, we chose to create our custom test environment. We prefered to use our own materials (e.g., resources) instead of using rented resources from existing CPs (e.g., Amazon EC2) for the following three reasons : 1) Most of the CPs including Amazon EC2 have restriction rules regarding any security testing and evaluating on their resources and systems [2]. 2) All large CPs list DoS attacks' testing and evaluating as a non-permissible action [2]. 3) No CP allows its users with direct access to the host. Therefore, gathering information (i.e. performance information) is a quite difficult task. Our testbed consists of three machines. One machine is used as a virtual machine host and the other machines are used as client and attack emulator. All machines are attached directly to a Linksys 1000 Mb/s SOHO switch. Our test network is completely disconnected from the network of our institution as well as from the Internet to avoid the leakage of the DoS attacks. The detection algorithm (Algorithm 2) is implemented in Python and the BoNeSi program is used [94] to generate attack-level and normal traffic. We used BoNeSi as a traffic generation tool because it allows us to simulate floods from large-scale bot networks. Moreover, BoNeSi tries to avoid the generation of packets with easily identifiable patterns, which can be quickly filtered out [94]. The virtual machine host used in the experiment is an Intel Core i7-4790 CPU 3.60 GHz Processor with 16 GB RAM. We installed the Apache 2.2 Web Server on the targeted VMs. The network interface is at 1000 Mb/s. We chose KVM [121] as our hypervisor-based virtualization. Indeed, KVM runs on the unmodified Linux kernel and is thus compatible with the standard performance tracing tools (e.g., LTTng), unlike Xen.

In order to simulate a real-world DoS attack, the CAIDA "DDoS Attack 2007" dataset [122] has been used as a baseline for extracting the features required to simulate attack traffic.

Since it is not possible to mine normal traffic data from CAIDA's dataset because it is collected at Darknet and has no normal traffic, we used another dataset to capture normal traffic. For this purpose, we used a traffic trace of a 5-minute non-flash-event activity before the first semi-final match of the 1998 FIFA World Cup' dataset [123]. Table 4.5 shows the traffic features and characteristics extracted from the CAIDA and 1998 FIFA World Cup' datasets (the same features used in [124]). We used then BoNeSi as a traffic generator to be run based on information given in Table 4.5.

Table 4.5 Attack and normal traffic features extracted from CAIDA and FIFA World Cup' datasets

Characteristic	**DoS Attacks**	**Normal Traffic**
Packet Size	64k	64k
No. of Sources	859	73
packet rate	125,705	385

4.5.2 Training Phase

During the generation of the attack and normal traffic using BoNeSi, we monitored the following metrics : CPU, memory, I/O and network load at different time intervals. We can also monitor and use high-level metrics, as illustrated in the trace abstractor section. However, we prefer to use these relatively basic metrics as they are widely used to describe the anomaly caused by a DoS attack [1]. The length of each interval is 30 seconds. For each interval, we used LTTng to generate the trace data. To extract CPU, memory, I/O and network loads from the trace data, we used the LTTng-analyses packages [125], which are a set of executable analyses to extract and visualise monitoring data and metrics from LTTng kernel traces. We created a training dataset that contains these performance metrics. The dataset is used then to train the SVM classifier to be able to distinguish between normal behavior and DoS attacks of the VM. We train an SVM classifier on our dataset using the 10-fold cross-validation model. We used a linear kernel function as it is considered more efficient for real-time applications [126]. It enjoys as well faster training and classification. Moreover, with the linear function, less memory is required as compared to the non-linear kernels [126].

4.5.3 Testing Phase

In order to test the proposed model in the presence of resources adjustments, we created a testing dataset that is suitable for each type of adjustments. To do this, we monitored the metrics (CPU, memory, I/O and network load) in the same way used to generate the training

dataset. However, in this phase, we performed some resources adjustments on the VMs during data collection, in order to study the effectiveness of our proposed model in such a case. We used the API of libvirt that employs cgroups [100] to adjust and limit VMs resources. Using cgroups allows exploiting Linux Kernel features that limit and allocate resources to VMs — such as CPU time, system memory, network bandwidth, or combinations of these resources [101]. This is performed on the two types of traffic datasets : attack and normal traffic. Our detection algorithm (Algorithm 2) is then executed on these datasets. The detection algorithm applies the filter of resources adjustment effect (obtained from executing Algorithm 1) on the testing dataset before passing it to the SVM classifier. The length of time used to determine the effect of resources adjustment in Algorithm 1 is 30 seconds.

We used to evaluate the accuracy of the proposed model the false positive, false negative, attack detection and accuracy rate.

Accuracy Rate =

$$100\% \times \frac{Total\ \#\ of\ correctly\ classified\ tuples}{Total\ \#\ of\ tuples} \tag{4.16}$$

Attack Detection Rate =

$$100\% \times \frac{Total\ \#\ of\ attacks}{Total\ \#\ of\ detected\ attacks} \tag{4.17}$$

False Positive Rate =

$$100\% \times \frac{Total\ \#\ of\ misclassified\ tuples}{Total\ \#\ of\ normal\ tuples} \tag{4.18}$$

False Negative Rate =

$$100\% \times \frac{Total\ \#\ of\ misclassified\ tuples}{Total\ \#\ of\ attack\ tuples} \tag{4.19}$$

The next section contains the results, compared to the traditional SVM and Decision-Tree detection [127] techniques. The traditional-SVM uses the SVM classifier directly, without applying any filtering strategy that can cope with the effect of the resources adjustment on the detection performance. Similarly, the Decision-Tree detection technique employs the Decision-Tree classification technique, without applying any filtering process. We train the traditional-SVM and Decision-Tree classifier on our dataset using the 10 fold cross-validation model.

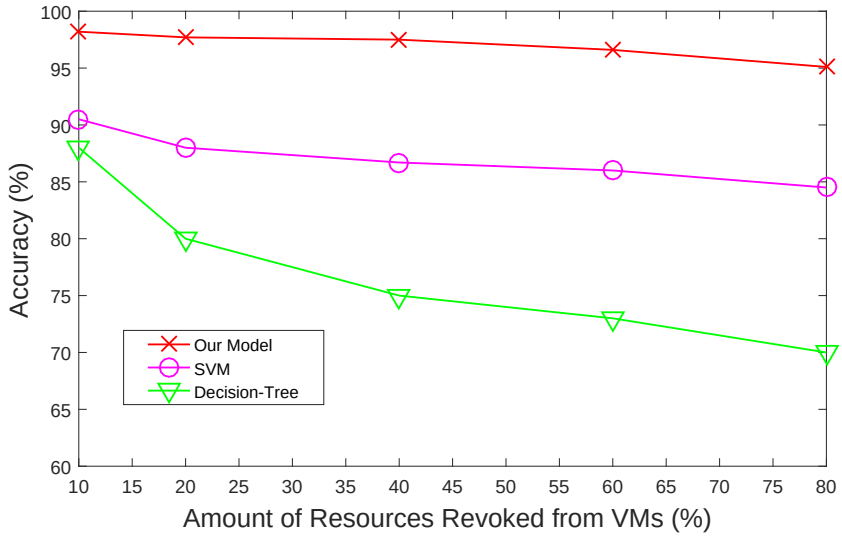


Figure 4.4 Accuracy with respect to (w.r.t.) amount of revoked resources

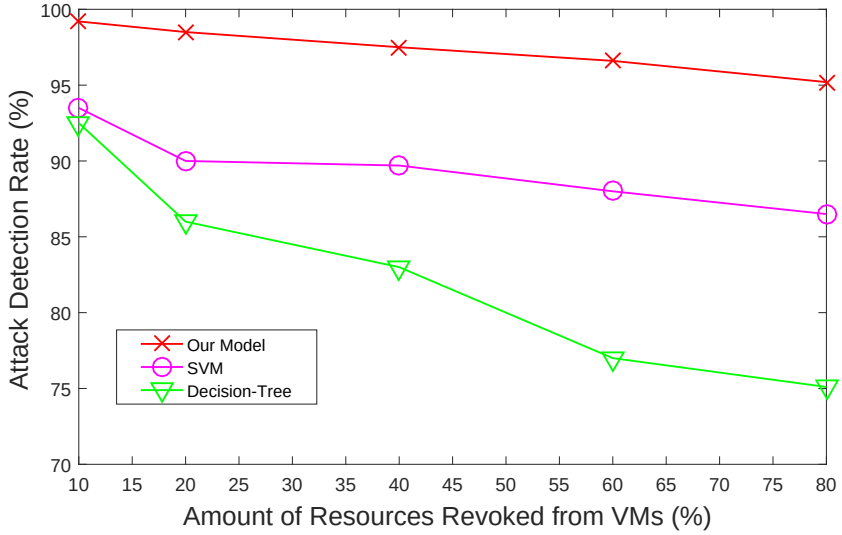


Figure 4.5 Attack detection rate w.r.t. amount of revoked resources

4.5.4 Experimental Results

We study in Figures 4.4, 4.5, 4.6, and 4.7 the performance of our framework with respect to the amount of resources revoked from VMs. The results reveal that our framework is resilient to the decrease in the VMs resources. More specifically, Figures 4.4 and 4.5 show respectively that the average accuracy and attack detection rates obtained by the proposed model at different percentages of revoked resources (from 10% to 80%) are 97.02% and 97.4%. These

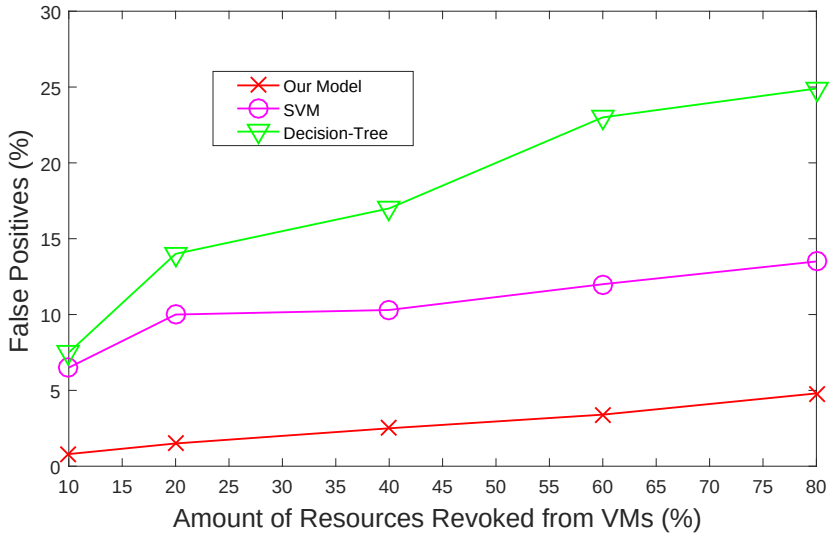


Figure 4.6 False positive percentage w.r.t. amount of revoked resources

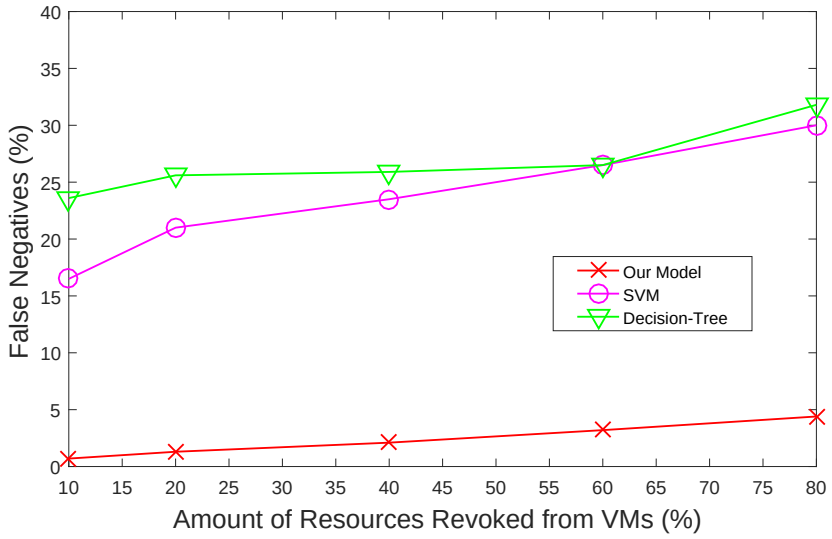


Figure 4.7 False negative percentage w.r.t. amount of revoked resources

results are better than the results obtained using the traditional-SVM (87.14% for accuracy and 89.54% for attack detection rate) and Decision-Tree (75.44% for the accuracy and 79.04% for attack detection rate). As for the false alarms, Figures 4.6 and 4.7 show respectively that the false positive and false negative rates obtained using our model at different percentages of revoked resources (from 10% to 80%) are 2.6% and 2.34%. These results are also better than the results obtained using traditional-SVM (10.46 % for the false positive and 23.5 % for the false negative) and Decision-Tree (17.28% for the false positive and 26.68% for the

false negative).

Moreover, Figures. 4.8, 4.9, 4.10, and 4.11 study the performance of our framework with respect to the amount of resources granted to VMs. These results reveal that our framework is also resilient to the increase in VMs resources. Figures 4.8 and 4.9 show respectively that the average accuracy and attack detection rate obtained by the proposed model at different percentages of granted resources (from 10 % to 80%) are 97.36% and 97.62%. These results are better than the results obtained using the traditional-SVM (80.66% for accuracy and 82.36% for attack detection rate) and Decision-Tree (75.84% for the accuracy and 77.56% for attack detection rate). As for the false alarms, Figures 4.10 and 4.11 show respectively that the false positive and false negative rates achieved by the proposed model at different percentages of revoked resources (from 10% to 80%) are 2.38% and 2.66%. These results are also better than the results obtained using the traditional-SVM (17.64 % for the false positive and 11.68 % for the false negative) and Decision-Tree (22.44 % for the false positive and 28.26% for the false negative).

The reason why our model performs better than the traditional-SVM and Decision-Tree approaches is that the proposed model takes into account the resources adjustments that occur in the VMs infrastructure. The detection algorithm (Algorithm 2) filters (using the filter obtained from Algorithm 1) the effect of resources adjustments in order to make the collected data (testing dataset) cope with the original infrastructure (on which the training was performed) before passing it to the SVM classifier. This is done by removing the effect of resources adjustments (Algorithm 2, line 5-9). This is unlike the traditional-SVM and Decision-Tree techniques, where the effect of the resources adjustments is totally ignored. Therefore, their accuracy for detecting DoS attacks is affected.

We calculate the percentage of accuracy that can be preserved using our model under changing infrastructure. To do so, we run our model without adjustments applied to the VMs resources. The accuracy of the detection obtained without adjustment was 99.40%. This result is used as a baseline to calculate the percentage of accuracy that our model can preserve under adjustments. For this purpose, we determine the amount of accuracy preserved under different amounts of adjustments and calculate the average, as in Table 4.6 for revoking-based adjustments and Table 4.7 for granting-based adjustments. The results show that the percentage of accuracy that can be preserved is 97.60 % for the revoking adjustments and 97.96% for the granting adjustments. This means that, by using our model, the accuracy got decreased under the effect of resources adjustments by only 1.79 % for the revoking adjustments and 1.43 % for the granting adjustments, which has no significant impact and can be neglected.

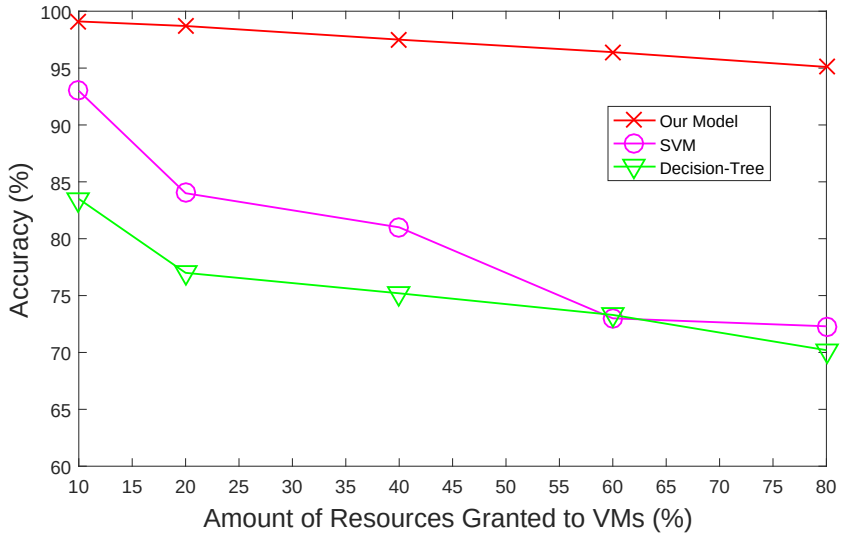


Figure 4.8 Accuracy w.r.t. amount of granted resources

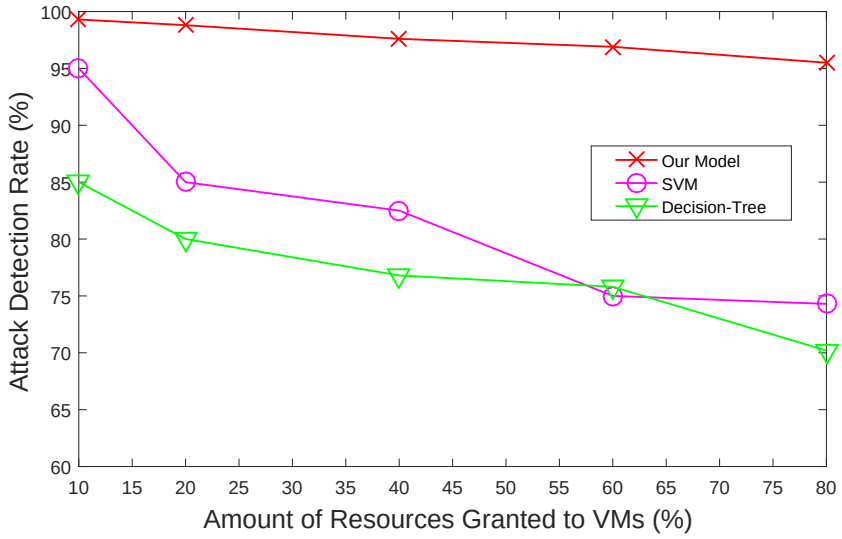


Figure 4.9 Attack detection rate w.r.t. amount of granted resources

It should be noticed also that the SVM kernel used in the experiments is the linear kernel. However, our results will not significantly change if we use another non-linear kernel (e.g., Quadratic kernel). In fact, we tested our detection model using different kernels and by considering different values of resources adjustment (granting and revoking adjustments from 10% to 80%). Table 4.8 shows the performance metrics obtained (the average result has been selected) for the different non-linear kernels. It presents the comparisons between different non-linear kernels using the proposed detection model. The results show that there is no

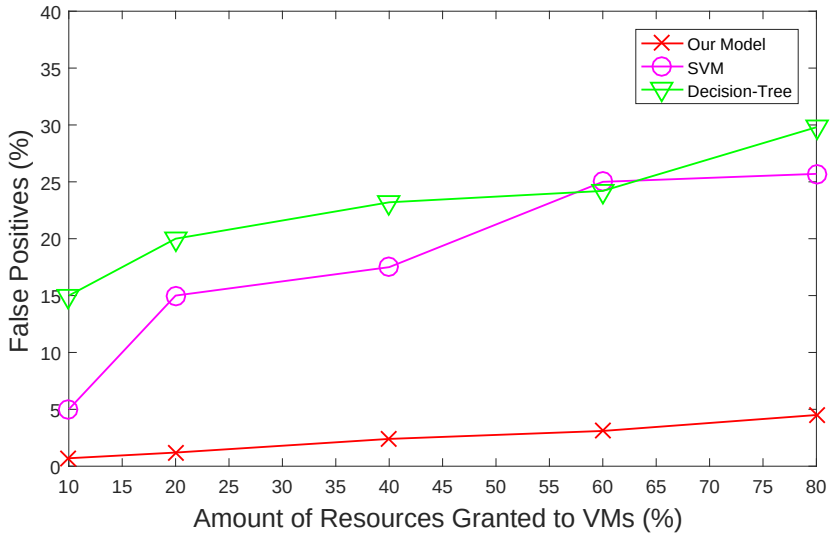


Figure 4.10 False positive percentage w.r.t. amount of granted resources

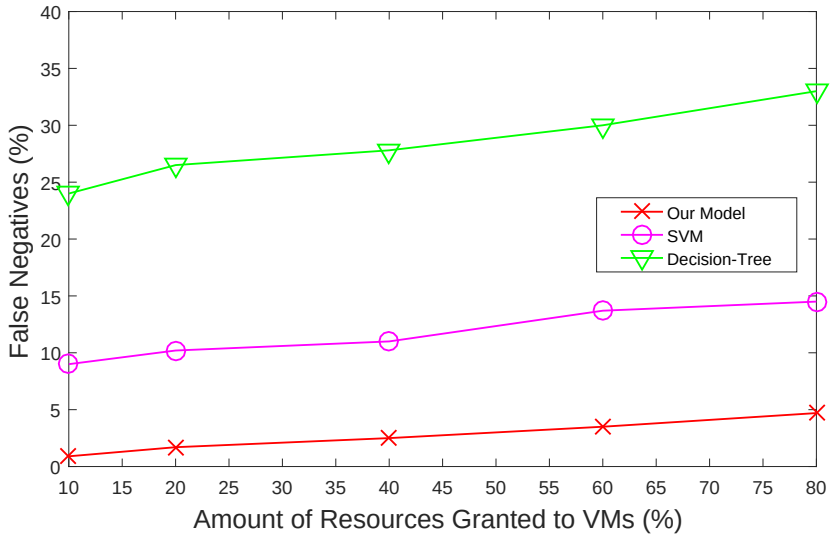


Figure 4.11 False negative percentage w.r.t. amount of granted resources

significant difference in the false positive, false negative, attack detection and accuracy rates between the different kernel functions.

Table 4.6 Amount of accuracy preserved by our model when revoking resources from VMs.

Resources revoked from VMs	**Accuracy preserved**
10 %	98.79 %
20 %	98.28 %
40 %	98.08 %
60 %	97.18 %
80 %	95.67 %
Average	97.60 %

Table 4.7 Amount of accuracy preserved by our model when granting resources to VMs.

Resources granted to VMs	**Accuracy preserved**
10 %	99.69 %
20 %	99.29 %
40 %	98.08 %
60 %	96.98 %
80 %	95.77 %
Average	97.96 %

Table 4.8 Kernel functions comparison using the proposed detection approach.

Kernel function	**Performance metric**			
	Accuracy (%)	Attack det. rate (%)	FPR (%)	FNR (%)
Linear kernel	97.19	97.51	2.49	2.50
Multilayer percepton kernel	97.03	97.09	2.90	2.92
Quadratic kernel	97.54	97.30	2.69	2.71
Polynomial kernel	97.51	98.23	1.99	2.11
Gaussian kernel	96.98	98.05	1.94	2.09

4.6 Conclusion

We present an SVM-based framework for detecting DoS attacks in a virtualized cloud under changing infrastructure. Our solution collects some system metrics to train the SVM classifier to be able to distinguish between normal and malicious (i.e., a DoS attack) activities of the VM. The hypervisor then monitors and quantifies the effect of performing resources adjustments on the collected data. This information is then used to maintain a filter of resources adjustments effect. The filter is used as a preprocessing step prior to classification to get rid of the noise that may show up on the collected data, and that may considerably decrease the accuracy of the detection. Moreover, our solution motivates VMs to declare their current business metrics to the hypervisor, to enable the hypervisor to correlate these metrics with the actual resources load and decide if it is coherent or not. This increases the possibility of identifying compromised VMs, trying to claim and consume more resources. Experimental results show that our model performs better than the traditional-SVM and Decision-Tree approaches in the presence of infrastructure adjustments, in terms of false positive, false negative, attack detection and accuracy rate. The results also show that the percentage of accuracy that can be preserved under resources adjustments using our model is 97.60 % for the revoking adjustments and 97.96 % for the granting adjustments. Our results also show that the accuracy got decreased under the effect of resources adjustments by only 1.79 % for the revoking adjustments and 1.43 % for the granting adjustments, which has no significant impact and can then be neglected.

CHAPTER 5 ARTICLE 2 : A TRUST-BASED GAME THEORETICAL MODEL FOR COOPERATIVE INTRUSION DETECTION IN MULTI-CLOUD ENVIRONMENTS

Adel Abusitta, Martine Bellaïche and Michel Dagenais

21st Conference on Innovation in Clouds, Internet and Networks (ICIN), IEEE, 2018.

Abstract

Cloud systems are becoming more complex and vulnerable to attacks. Cyber attacks are also becoming more sophisticated and harder to detect. Therefore, it is increasingly difficult for a single cloud-based intrusion detection system (IDS) to detect all attacks, because of limited and incomplete knowledge about attacks. The recent researches in cyber-security have shown that a cooperation among IDSs can bring higher detection accuracy in such complex computer systems. Through collaboration, a cloud-based IDS can consult other IDSs about suspicious intrusions and increase the decision accuracy. The problem of existing cooperative IDS approaches is that they overlook having untrusted (malicious or not) IDSs that may negatively effect the decision about suspicious intrusions in the cloud. Moreover, they rely on a centralized architecture in which a central agent regulates the cooperation, which contradicts the distributed nature of the cloud. In this paper, we propose a framework that enables IDSs to distributively form trustworthy IDSs communities. We devise an algorithm, based on cooperative game theory, that allows a set of cloud-based IDSs to cooperatively set up their coalition in such a way to make their individual detection accuracy increase, even in the presence of untrusted IDSs.

5.1 Introduction

Cloud-based cyber-attacks are known to be more complex and harder to detect. It became significantly more difficult for a traditional single intrusion detection system, whether it is network-based, hypervisor-based, or VM-based, to detect all attacks, due to limited knowledge about attacks. Collaboration among intrusion detection systems (IDSs) can be used to gain higher detection accuracy as compared to traditional single IDS. Through collaboration, IDSs in different regions, and possibly, belonging to different Cloud Providers (CPs) can cooperate in such a way that makes them utilize the expertise of each other to cover and identify unknown attack patterns.

A cloud-based IDS can be classified into two types ; signature-based and anomaly-based [1]. The former compares suspicious behavior with known attack patterns. In order to make signature-based effective, the signature database should be updated frequently. On the other hand, anomaly-based IDS raises alarms when unusual and/or unexpected observations are detected. Anomaly-based IDSs are effective to detect unknown attacks. Moreover, they do not need a database of known attacks. However, the shortcoming of using anomaly-based detection is the relative high false positive rate compared to the signature-based technique [11]. IDSs may adopt both techniques to have improved detection accuracy. However, the detection accuracy is limited by the amount of knowledge they have (e.g., their security vendors have). Recent research [11] [23] shows that the collaborative detection can enhance the detection rate up to 60%. Through collaboration, an IDS can benefit from other IDSs expertise by consulting them about suspicious behavior. The feedback received can be then used to decide whether to rise an alarm or not.

The main limitation of existing cloud-based cooperative IDS (e.g. [63] [64] [65] [66] [128] [129]) is that they work under the assumption that all IDSs are trustable, which lets their collaboration systems vulnerable to untrusted (malicious or not) insiders.

To address the aforementioned problems, we propose a trust-based framework for cooperative IDS in a multi-cloud environment. The framework can be summarized as follows. We enable an IDS to evaluate other IDSs' trustworthiness. This is done by considering its personal experience using bayesian inference. After obtaining IDSs' trust values, a coalition formation algorithm is used, that is based on the coalitional game theory [130]. The algorithm enables IDSs to leave or join a given coalition in such a way that enhances its chance to work with trusted IDSs. Thereafter, we propose a feedback aggregation algorithm, that is based on the Dempster-Shafer Theory (DST) [16], to enable an IDS inside a coalition to aggregate feedbacks from different IDSs about suspicious intruders, which helps make the optimal decision in terms of detection accuracy.

Unlike similar proposals (e.g. [67]), we adopt a distributed approach in which each IDS autonomously makes its own decisions. This, in turn, avoids the difficulty of finding a third party that is trusted by all the IDSs. Also, it reduces the instability inside the coalition due to a single point of failure. In summary, our work consists of the following contributions :

— Modeling and proposing a framework that enables cloud-based IDSs to distributively form trustworthy IDS communities. More specifically, we present a systematic approach that considers the trustworthiness of IDSs through creating cooperative IDS.
— Proposing a new trust evaluation approach, based on Bayesian inference, that enables a cloud-based IDS to evaluate another IDS's trustworthiness based on its personal

experiences.

— Devising an algorithm, based on cooperative game theory, that allows a set of cloud-based IDSs to cooperatively set up their coalition in such a way to increase their individual detection accuracy in the presence of untrusted IDSs. The proposed algorithm converges to a Nash-stable situation ; that is, no IDS has an incentive to leave its current coalition to move to another coalition.

The rest of this paper is organized as follows. In Section 5.2, we discuss the related work. We present the trust-based cooperative intrusion detection system in Section 5.3. In Section 5.4, we present our empirical results to show the effectiveness of the proposed approach. Finally, Section 5.5 concludes the paper.

5.2 Related Work

Cloud-based cooperative IDSs have been proposed in many works in the past. For example, Lo et al. [62] propose a cooperative detection approach within the cloud computing environment. Alerts are exchanged between the cloud environment nodes (i.e., hosts) whenever an attack gets detected. They use a rule-based technique to detect TCP SYN attacks by fetching the threshold for rule patterns through the initial rule establishment phase. The main advantage of this approach is that it is able to distribute the detection overhead between the cloud nodes. Recently, Teng et al. [61] proposed an approach that combines two detectors : a feature detector and a statistical detector. The feature detector uses SNORT to separate events based on network protocols (e.g., TCP). The statistical detector cooperates with the feature detector by using data packets from it to determine whether an event is an attack or not. If the rate of packets obtained exceeds the predefine threshold, then this case will be considered as an attack.

Man and Huh [63] and Singh et al. [64] proposed a cooperative IDS between cloud computing regions. Their method allows exchanging alerts from multiple elementary detectors. In addition, they enable the exchange of knowledge between interconnected clouds. Ghribi [65] proposed a middleware IDS. The approach enables a cooperation between cloud IDS layers : Hybervisor-based IDS, Network-based IDS and VM-based IDS. If an attack is detected in a layer, the attack cannot be executed in the other layers. Chiba et al. [66] introduced a cooperative network-based cooperative intrusion detection system to identify network attacks in the cloud environment. This can be done by monitoring network traffic while maintaining performance and service quality.

The main limitation of the above mentioned approaches is that they work in the assumption

that all IDSs are trustable, which makes their collaboration systems vulnerable to malicious insiders. The aim of this paper is to present a systematic approach to build a cloud-based cooperative IDS that adopts trust assessment mechanisms and supports trustworthy aggregation decisions. The proposed approach should work in the presence of untrusted cloud-based IDSs .

In a multi-cloud environment, Dermott et al. [67] proposed a cooperative intrusion detection in federated cloud environments. They use the Dempster-Shafer theory of evidence to collect the beliefs provided by the watching entities. The collected beliefs are used to make the final decision regarding a possible attack. The main limitation of this approach is that it is based on a centralized architecture, whereby a trusted third-party called broker is responsible for collecting feedback and managing intrusion detection.

In a non-cloud environment, a cooperative IDS has also been recently proposed in [68] [69] [5] [6] [7] [8] [9] [10]. However, these works also have the limitation of the above mentioned approaches, since they rely on the assumption that all IDSs are trustable, which makes their collaboration system vulnerable to malicious insiders.

A trust-based cooperative IDS has been proposed in a non-cloud environment. For example, Fung and Zhu [11] present a trust-based collaborative decision framework. Through cooperation, a local intrusion detection system (IDS) can detect new attacks that may be known to other IDSs, which may be from different security vendors. They study how to utilize the diagnosis from different IDSs to perform intrusion detection. They present a system architecture of a collaborative IDS in which trustworthy feedback aggregation is a key component. Similarly, Zhu et al. [70] [71] proposed an incentive-based communication protocol, which provides IDS nodes incentives to send feedbacks to their peers, and thus to prevent malicious behaviors. The main limitation of the existing trust-based cooperative IDS is that it considers a consultation request to be sent to many IDSs in order to get a feedback. This in turn causes extra overhead, through consulting needlessly some IDSs (i.e., untrusted IDSs). This is unlike our approach, where we use a coalitional game, in order to construct a set of trusted IDSs and thus minimise the number of consultation requests while guaranteeing higher detection accuracy.

In general, for a multi-cloud environment, a decentralized framework that considers trustworthiness of IDSs during the cooperation had yet to be addressed. Thus, in this paper, we present a trust-based cooperative IDS in a multi cloud environment. This in turn, enhances the detection accuracy compared to the existing cooperative and non-cooperative IDSs.

5.3 The Proposed Trust-based Cooperative IDS

In this section, we present a trust-based cooperative IDS in a multi-cloud environment. The section is divided into the following subsections : trust evaluation, trust-based coalition formation algorithm and feedback aggregation (Fig. 5.1).

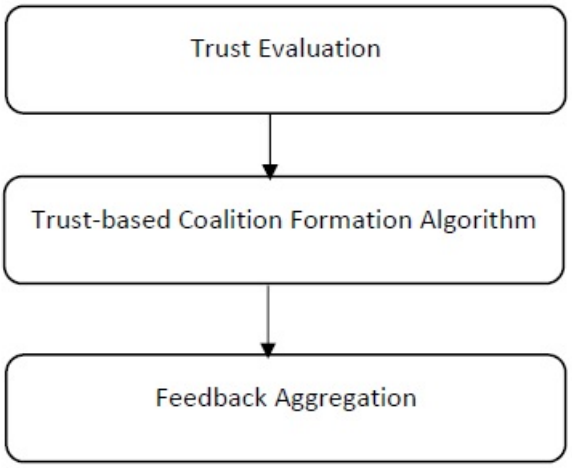


Figure 5.1 Proposed Methodology

5.3.1 Trust Evaluation

A cloud-based IDS can evaluate the trust value of another IDS based on its personal experience with that IDS. We adopt a Bayesian inference approach to compute the trust value of an IDS [14]. The Bayesian inference was chosen because it is well-founded to derive trust values [131]. When the cloud-based IDS consults another IDS regarding a suspicious intruder, the received feedback and the revealed result (i.e., attack or not) are used to update the trust value of the consulted IDS. The trust value can be promoted if the IDS successfully diagnosed the consultation request about a suspicious intruder and it can be demoted otherwise. The trust value here represents and shows the accuracy of the IDS diagnosing suspicious attacks. An IDS $i \in \mathcal{N}$, where $\mathcal{N}$ is a set of IDSs, is endowed with a belief function, which computes the trust level of another IDS $j \in \mathcal{N}$. The new trust value t'_j is derived from the old trust value t_j as follows :

$$t'_j = F(t_j; \alpha_j, \beta_j) \tag{5.1}$$

where F is the regularized incomplete beta function [14], which is also the cumulative beta

distribution function of the following beta probability density function :

$$f(x;\alpha_j,\beta_j) = \frac{x^{\alpha_j-1}(1-x)^{\beta_j-1}}{\mathbf{B}(\alpha_j,\beta_j)} \tag{5.2}$$

B represents the complete beta function. The value of α_j and β_j are updated after receiving the feedback from an IDS j. β_j is increased when the IDS j successfully diagnoses the consultation request. Eq. (5.4) describes the update of β_j.

$$\beta_j = \beta_j \times (1+\rho_j) \tag{5.3}$$

where ρ_j represents the weight of the diagnosed consultation request if it is successful and 0 if not.

Eq. (5.4) describes the update of α_j.

$$\alpha_j = \alpha_j \times (1+\gamma_j) \tag{5.4}$$

where γ_j denotes the weight of the diagnosed consultation request if it is unsuccessful and 0 if not.

The values of ρ_j and γ_j should be carefully set by an IDS i who is requesting feedback about a suspicious attack from other IDSs. These values reflect the detection difficulty degree of the suspicious intruder. The higher value of β_j will increase the trust of an IDS j while a higher value of α_j will decrease it.

The initial trust value t_j is obtained at the beginning during the testing period. The total reported diagnosis data from peer IDS j is denoted by the set $\mathcal{M}_j$. The initial trust value represents the total number of consultation requests that have been successfully diagnosed over the total number of consultation requests :

$$t_j = \frac{\sum_{k\in\mathcal{M}_j} r_{j,k}}{|\mathcal{M}_j|} \tag{5.5}$$

Where the parameter $r_{j,k}$ is the revealed result of the k-th diagnosis request : $r_{j,k}$ =1 indicates successful diagnosis of the k-th request. $r_{j,k}$ =0 indicates otherwise.

The initial value of α and β can be obtained as follows :

$$\alpha_j = \sum_{k\in\mathcal{M}_j} (1-r_{j,k}) \tag{5.6}$$

$$\beta_j = \sum_{k \in \mathcal{M}_j} (r_{j,k}) \tag{5.7}$$

5.3.2 A Trust-based Coalition Formation

In this section, we model the problem of coperative IDS as a coalition formation cooperative game with non-transferable utility [95].

5.3.2.1 Characterization

The proposed coalition formation algorithm is a hedonic coalitional game [95], [15] [132] [103], a category of coalition formation games [130], [15], [133] in which each agent (i.e. IDS) acts selfishly, and its preferences for a coalition depend only on the members of that coalition. A hedonic game is used due to the fact that finding the optimal coalition structure, in coalition formation, is NP-complete [134]. Therefore, a hedonic game, which satisfies stability features was used. Stability indicates that none of the coalition members (i.e. IDSs) finds an incentive to leave its current coalition and join another one.

To establish the model, we need to define a preference relation so that each IDS can order and compare all the possible coalitions it belongs to and build preferences over them. For any IDS $i \in \mathcal{N}$, where $\mathcal{N}$ is a set of IDSs, a preference relation $\succ_i$ is defined as a transitive binary relation over the set of all coalitions that IDS i can form [95]. Specifically, for any IDS $i \in \mathcal{N}$, and given two coalitions C_1, C_2, the notation $C_1 \succ_i C_2$ means that IDS i prefers being a member of C_1 rather than C_2.

In our coalition formation game, the preference function of the IDSs can be defined as follows :

$$C_1 \succeq_i C_2 \Longleftrightarrow f_i(C_1) \geq f_i(C_2) \tag{5.8}$$

where C_1, $C_2 \subseteq \mathcal{N}$ are two coalitions containing IDS i, and $f_i(.)$ is a preference function defined as follows :

$$f_i(C_k) = U_i(C_k) = \prod_{j \in C_k} T_i^j \tag{5.9}$$

$\prod_{j \in C_k} T_i^j$ is denoted as the coalition trust criterion. T_i^j is denoted as IDS i beliefs in IDS j $\in \mathcal{N}$.IDS i's beliefs in C_k's members is obtained using Bayesian inference as in Eq. (5.1). We use the product of IDSs trust values instead of their summation in the definition of the coalition trust criterion in order to conserve the effect of small trust values on the global coalitions trust value. That way, the impact of a small trust value will not be mitigated by a higher one.

5.3.2.2 The Proposed Coalition Formation Algorithm

The algorithm (Algorithm 4) that we propose is based on the hedonic shift rule [95] : let $\Pi = \{C_1, ..., C_l\}$ represent the set of coalition partitions. That is, for k = {1, 2, . . . , l}, each $C_k \subseteq \mathcal{N}$ is a disjoint coalition. Each IDS $i \in \mathcal{N}$ decides to leave its current coalition $C_\Pi(i)$ to join another one $C_k \in \Pi \cup \phi$ if and only if its coalition trust criterion (i.e., $U_i(C_k) = \prod_{j \in C_k} T_i^j$) in the new coalition exceeds the one it obtains in its current coalition. Leaving and joining decisions are considered selfish decisions. This means that they are made without considering their effect on the other IDSs. In Algorithm 4, an IDS i evaluates all of the possible coalitions

Algorithm 4: Trust-based Coalition Formation Algorithm

Given the current coalition partition $\Pi_c = \{C_1, ..., C_l\}$, each IDS i evaluates possible shift from its current coalition as follows :

repeat
- **foreach** $C_k \in \Pi_c \cup \phi$ **do**
  - **foreach** *IDS* $j \in C_k$ **do**
    - — calculate the trust value of IDS j.
  - **end**
- **end**
- calculate $U_i(C_k \cup \{i\})$ and $U_i(C_{\Pi_c}(i))$
- **if** $U_i(C_k \cup \{i\}) > U_i(C_{\Pi_c}(i))$ **then**
  - — IDS i leaves its current coalition $C_{\Pi_c}(i)$ and joins the new coalition.
  - — Π_c is updated : $\Pi_{c+1} = (\Pi_c \setminus \{C_{\Pi_c}(i), C_k\}) \cup \{C_{\Pi_c}(i) \setminus \{i\}, C_k \cup \{i\}\}$.
- **else**
  - — IDS i remains in the same coalition so that : $\Pi_{c+1} = \Pi_c$
- **end**

until ε *elapses*;

it can join or form, beginning by leaving its current coalition C_{Π}(i) to join another already existing coalition C_k. The algorithm computes the trust value for each IDS $j \in C_k$ as in (1). Then, the algorithm determines the coalition trust criterion $U_i(C_{\Pi}(i))$ of its current coalition C_{Π}(i) as in (9) and compares it with the coalition trust criterion $U_i(C_k)$ of the coalition C_k. If the coalition trust criterion of the current coalition is greater than that of the coalition C_k, then the IDS i leaves its current coalition to join C_k. Otherwise, IDS i remains in its current coalition. One should note that, after a certain fixed period of time ε, the whole process is repeated, in order to obtain the changes that may happen in the current coalition partition Π_c. These changes include changes in the IDSs trust values, the departure of existing IDSs and the arrival of new IDSs.

The main complexity of Algorithm 5 lies in the shifting operations, i.e. the process of finding a new coalition to join, which equals $O(|\Pi_c|)$, where $|\Pi_c|$ is the number of coalitions in the current coalition partition.

The algorithm can be implemented in a distributed manner, given that each IDS can act autonomously and independently from any other IDSs in the system. However, it is important to provide appropriate actions based on [22] for :

— State recovery : the algorithm assumes that each IDS is able to retrieve the current coalition partition Π_c. Any state retrieval algorithm available in the state-of-the-art (e.g. [135], [136]) can be used for this purpose ;
— Atomic state update : to guarantee correctness, Π_c must not change while IDS i moves from its current coalition C_{Π}(i) and joins another one. Distributed mutual exclusion algorithms (e.g. [137]) can be used for this purpose.

5.3.3 Feedback Aggregation

In the previous section, we presented a trust-based coalition formation model that enables a set of cloud-based IDSs to cooperatively set up their coalitions. The output of the coalition formation algorithm (Algorithm 4) is a set of coalitions, where each coalition consists of a set of IDSs that prefer to work with each other. In this section, we show how an IDS inside a coalition can aggregate feedbacks received from other IDSs in the same coalition. For this purpose, we use the Dempster-Shafer Theory (DST) for feedback aggregation. DST was selected for the following main reasons : (1) unlike other aggregation models (e.g. Bayesian aggregation model) that demand complete information of prior probabilities, DST can handle a lack of complete information (i.e. uncertainty), and (2) it has a property to prevent collusion attacks, which occur when several malicious IDSs collaborate to give misleading judgments.

In our model, the frame of discernment, which describes the status of a suspicious intruder is Ω= $\{1, 0, U\}$ denotes a set consisting of three hypotheses. 1 means that IDS j decides and reports to IDS i that there is an intrusion, 0 means that IDS j decides and reports to IDS i that there is no intrusion, and U shows that IDS j is uncertain whether there is an intrusion or not.

DST combines multiple IDSs beliefs under the condition that evidences from different IDSs are independent. For example, if IDS i wants to combine the belief of two IDSs IDS_1 and IDS_2 over the same frame of discernment Ω, the combined belief of IDS_1 and IDS_2 is calculated as follows [138] :

$$\begin{aligned} &m_{IDS_1}(1) \oplus m_{IDS_2}(1) = \frac{1}{K}[m_{IDS_1}(1)m_{IDS_2}(1)+ \\ &m_{IDS_1}(1)m_{IDS_2}(U) + m_{IDS_1}(U)m_{IDS_2}(1)] \end{aligned} \tag{5.10}$$

$$\begin{aligned} &m_{IDS_1}(0) \oplus m_{IDS_2}(0) = \\ &\frac{1}{K}[m_{IDS_1}(0)m_{IDS_2}(0) \\ &+ m_{IDS_1}(0)m_{IDS_2}(U) + m_{IDS_1}(U)m_{IDS_2}(0)] \end{aligned} \tag{5.11}$$

$$m_{IDS_1}(U) \oplus m_{IDS_2}(U) = \frac{1}{K}[m_{IDS_1}(U)m_{IDS_2}(U)] \tag{5.12}$$

where,

$$\begin{aligned} K = &m_{IDS_1}(1) + m_{IDS_2}(1) + m_{IDS_1}(1) + m_{IDS_2}(U) \\ &+ m_{IDS_1}(U) + m_{IDS_2}(U) + m_{IDS_1}(U) + m_{IDS_2}(1) \\ &+ m_{IDS_1}(U) + m_{IDS_2}(0) + m_{IDS_1}(0) + m_{IDS_2}(0) \\ &+ m_{IDS_1}(0) + m_{IDS_2}(U) \end{aligned} \tag{5.13}$$

Here is an example. Assume the following :
$m_{IDS_1}(1) = 0.75$ $m_{IDS_1}(0) = 0$ $m_{IDS_1}(U) = 0.25$
$m_{IDS_2}(1) = 0.6$ $m_{IDS_2}(0) = 0$ $m_{IDS_2}(U) = 0.4$

by combining the above two belief functions, we can obtain the result as follows :
$belief(1) = (0.75 * 0.6) + (0.75 * 0.4) + (0.6 * 0.25) = 0.9$
$belief(0) = (0 * 0) + (0 * 0.4) + (0 * 0.25) = 0$
$belief(U) = (0.25 * 0.4) = 0.1$

Since $belief(1) > belief(0) > belief(U)$, IDS i will decide that an attack exists.

5.4 Experimental Evaluation

In this section, we first explain the experimental setup used to perform our experimentation and then study the performance of the proposed cooperative intrusion detection approach.

5.4.1 Experimental Setup

We implemented our framework in a 64-bit Windows 8 environment on a host equipped with an Intel Core i7-4790 CPU 3.60 GHz Processor and 16 GB RAM. We used Matlab for implementing our model.

The simulation environment uses 100 cloud-based IDSs. Each IDS is represented by two parameters, trust value t and decision threshold τ. The trust value represents the expertise level of the IDS, which in turn represents the ability of the IDS to catch suspicious traces from a given observation. The threshold τ represents the sensitivity of the IDS. Lower values of τ indicate a more sensitive IDS.

We use a Beta density function to reflect the intrusion detection capability of each IDS. A Beta density function is given by :

$$\begin{aligned} f(z|\alpha,\beta) &= \frac{1}{B(\alpha,\beta)z^{\alpha-1}(1-z)^{\beta-1}} \\ B(\alpha,\beta) &= \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}dx \end{aligned} \tag{5.14}$$

$$\begin{aligned} \alpha &= 1 + \frac{t(1-d)}{d(1-t)}r \\ \beta &= 1 + \frac{t(1-d)}{d(1-t)}(1-r) \end{aligned} \tag{5.15}$$

where $z \in [0, 1]$ is the assessment result from the IDS about the likelihood of intrusion, and $f\ (z|\alpha,\beta)$ is the distribution of assessment z from an IDS with trust level t to an intrusion with difficulty level $d \in [0, 1]$. The trust level in the distribution can represent the expertise level of the IDS. Higher values of d represent these attacks that are difficult to detect. Higher values of t indicate a higher probability of generating correct intrusion assessments. $r \in \{0, 1\}$ is the expected result of detection. $r = 1$ means that there is an intrusion and $r = 0$ means otherwise.

In order to evaluate the ability of the proposed model in the presence of an untrusted en-

vironment, We made the percentage of untrusted IDSs 70% (trust level $t \leq 0.2$). We argue, based on the recent literature [139], that the percentage of untrusted nodes tends to form the majority compared to that of trusted nodes. We applied the proposed coalition formation algorithm (Algorithm 4) on the considered IDSs. We compared the proposed aggregation approach with other known aggregation approaches in the state-of-the-art : Majority aggregation model [62] and the weighted average aggregation model [96]. In the majority model, the IDS collects feedback from IDSs about suspicious behaviour and the decision is made (i.e., attack or not) according to the majority. However, in the weighted average aggregation model, weights W are assigned to feedbacks from different IDSs to distinguish their detection capability. Highly trusted IDSs are assigned with larger weights compared to low trusted IDSs. The decision is made according to the following equation. If $(\Sigma_{k=1}^{n}\ W_k y_k)$ / $(\Sigma_{k=1}^{n}\ W_k) \geq \tau$, the decision is *the existence of an attack*. Otherwise, the decision is that *there is no attack*, where W_k is the weight of the k-th IDS and y_k is the feedback from the k-th IDS.

5.4.2 Experimental Results

In Fig. 5.2, we observe that the proposed aggregation model (i.e., Dempster-Shafer aggregation approach) shows significant improvement for the false negative rate, compared to the weighted and majority aggregation model at different threshold values τ. Similarly, in Fig. 5.3, our model yields significant improvement for the false positive rate, compared to the other two models. This is justified by the fact that Dempster-Shafer disregards the untrustworthy feedbacks upon building the final decisions. Moreover, Dempster-Shafer gives a weight for each feedback according to the trustworthiness level of the IDS giving this feedback.

In Fig. 5.4 and Fig. 5.5, we also study the effect of the trust value (i.e., expertise level) on the accuracy of the detection. To this end, we run our Algorithm (Algorithm 4) many times. Each time, we let IDSs have different values of t. The study is conducted at different threshold values τ. Fig. 5.4 shows that the false negative decreases when the trustworthiness level of an IDS increases. Similarly, Fig. 5.5 shows that the false positive decreases when the trustworthiness level of an IDS increases. This is justified by the fact that whenever an IDS becomes more trusted, it will be able to give a right feedback about suspicious attacks.

Fig. 5.6 gives a comparison between the proposed trust-based coalitional game approach and the trust-based grand coalition approach. The latter considers all existing IDSs during the cooperation. In other words, the coalition is done among all IDSs. Thus, the feedback is received from all IDSs and the final decisions are made using the same proposed aggregation model (i.e., Dempster-Shafer). This is unlike our approach where we first run a coalition

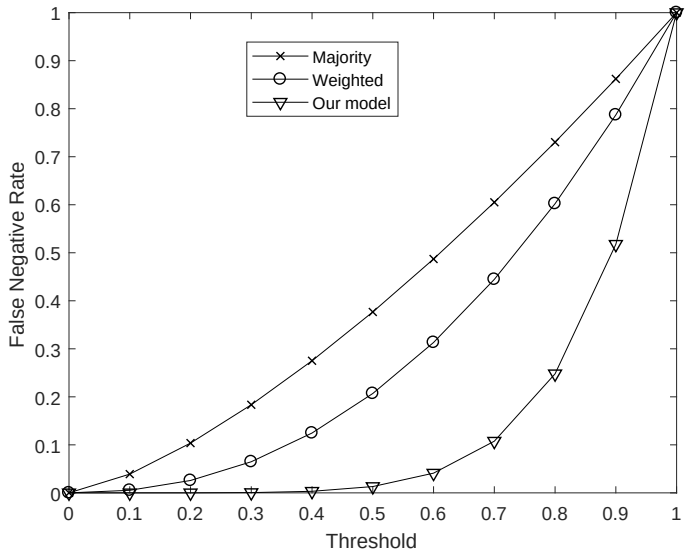


Figure 5.2 Comparison of three aggregation models (False Negative Rate).

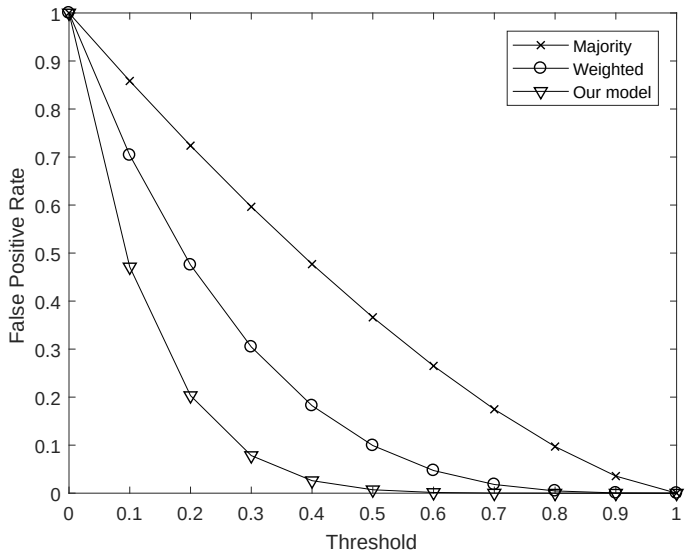


Figure 5.3 Comparison of three aggregation models (False Positive Rate).

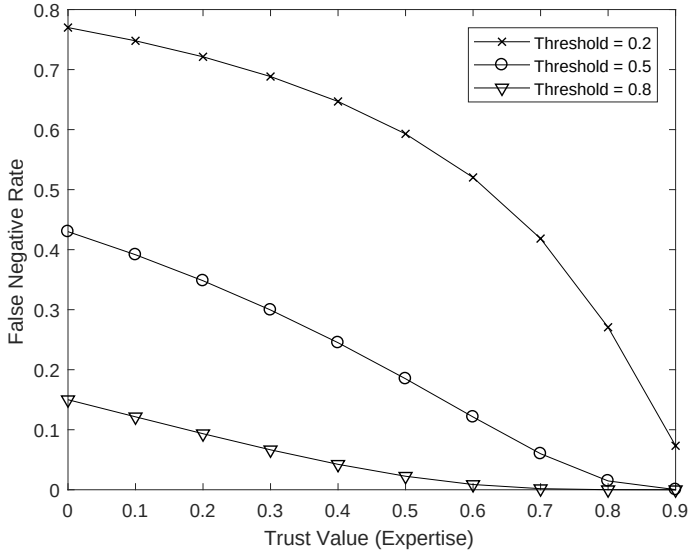


Figure 5.4 False Negative vs. Trust Value t .

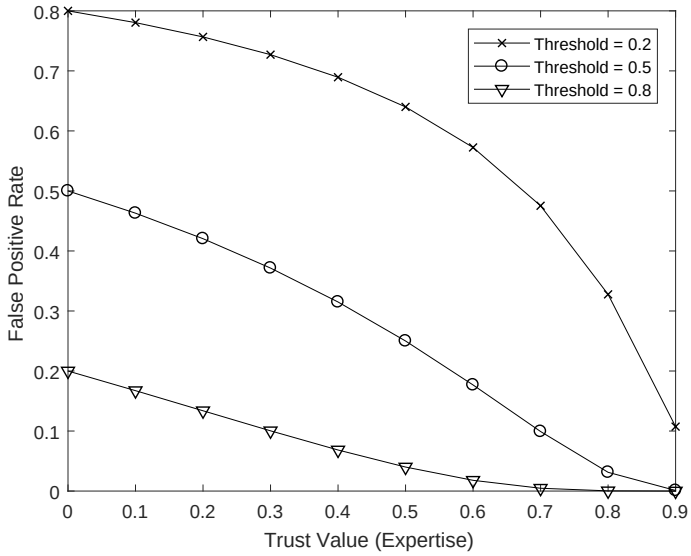


Figure 5.5 False Positive vs. Trust Value t.

formation Algorithm (Algorithm 4) and minimise the number of IDSs inside the coalition. The figure shows the superiority of the proposed model for both the false positive and false negative rates. This is due to the fact that the proposed coalition approach minimises the number of untrusted IDSs inside each generated coalition. Thus, the received feedback is more likely to reflect the real status of any suspicious behaviour, whether it is a real attack or not. However, for the grand coalition approach, the feedback is received from every existing IDS. Therefore, there will be a chance of receiving incorrect feedback from untrusted IDSs. Fig. 5.6 also studies the cost associated with using each approach. The cost represents the time needed to make a judgment about a suspicious attack. The result is projected in a range between 0 and 1. Our model yields a minimum overhead compared to the grand coalition approach. The reason is that our model minimises unnecessary consultation requests by consulting only those trusted IDSs in the final coalition. This is unlike the grand coalition approach where a consultation request is sent to all existing IDSs.

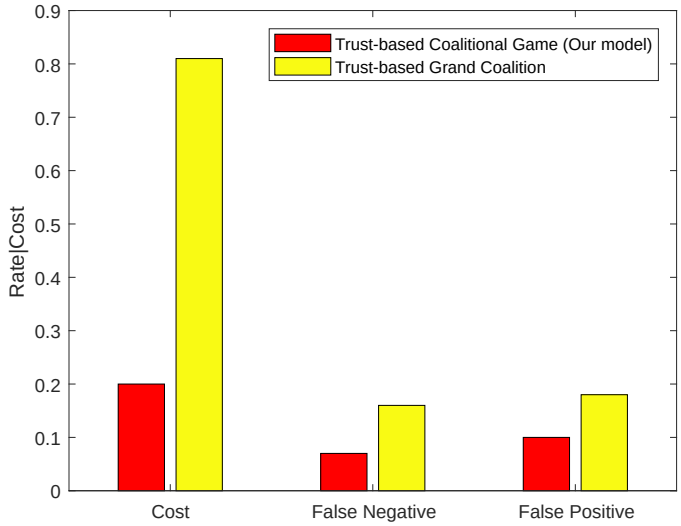


Figure 5.6 Comparison of two coalition formation models.

5.5 Conclusion

This paper investigates a novel trust-based cooperative IDS in a multi-cloud environment. To this end, we propose a cooperative game-theoretic framework. The framework enables an IDS to evaluate the trust value of other IDSs using bayesian inference. We devise a coalition formation algorithm, that is based on the coalitional game theory. The algorithm enables IDSs to leave or join a given coalition in such a way that enhances their ability to work with

trusted IDSs. The proposed algorithm converges to a Nash-stable situation ; that is, no IDS has an incentive to leave its current coalition to move to another coalition. Furthermore, we propose a feedback aggregation algorithm, that is based on Dempster-Shafer Theory (DST), to enable an IDS inside a coalition to aggregate feedbacks about suspicious attacks in order to make the optimal decision in terms of detection accuracy. Numerical results show the effectiveness of the proposed approach in terms of false positive and false negative rates, and cost.

CHAPTER 6 ARTICLE 3 : ON TRUSTWORTHY FEDERATED CLOUDS : A COALITIONAL GAME APPROACH

Adel Abusitta, Martine Bellaïche and Michel Dagenais
Computer Networks, vol. 145, pp. 52-63, 2018.

Abstract

The demands for cloud-based applications are expected to increase in such a way that current Cloud Providers' (CPs) resources may become insufficient. This promotes the need to outsource some of the requested Virtual Machines (VMs) to other CPs. A Cloud Federation (CF) provides an effective platform that enables CPs to upgrade their resource scaling strategies. Several CF approaches have been proposed, but they suffer from the hazard of working with untrusted (malicious or not) CPs, resulting in performance degradation. To address this problem, we introduce a trust-based framework for CF formation. Our model enables a CP to evaluate other CPs' trustworthiness by considering two approaches : objective and subjective trust evaluations. In the former, Bayesian inference is used to compute trust values based on previous interactions. In the latter, the Dempster-Shafer Theory (DST) integrated with the Bayasian inference is used to compute trust values in the absence of previous interactions. Thereafter, a novel decentralized algorithm is devised, based on coalitional game theory, that allows heterogenous CPs to establish their coalitions in such a way that maximises the trust of the formed federations. Experimental results show that our proposed algorithm enhances throughput, response time and availability of federated CPs compared to the QoS-based and Grand formation models.

6.1 Introduction

Cloud Computing enables Cloud Providers (CPs) to rent out space on their infrastructures, platforms and services to many consumers. This becomes possible thanks to virtualization that enables the easy migration of applications and services from one node to another. Many companies, organizations and governments are expected to transfer, if they have not already, all or parts of their IT solutions to the cloud [99] [140]. This transfer is profitable from an economic point of view since it allows them to streamline technology infrastructure expenses and capital costs.

One of the main issues that must be faced by CPs, due to the huge demands on their services, is the problem of insufficient resources to fulfill the requested VMs. This promotes the need for CPs to delegate these requests to other CPs in order to upgrade their resource scaling capabilities. A Cloud Federation (CF) provides an effective platform to address the aforementioned challenges [20]. The purpose of the CF consists of grouping CPs to fulfill the dynamic resource requests of users/applications to support data-intensive workloads [21]. Thus, through the use of CFs, CPs can benefit from each other's resources to run the VMs [22] [20] [21], in order to improve individual performance and enhance users' satisfaction.

Existing works in CF (e.g. [74] [21] [75] [76] [83]) significantly focus on forming the federation among CPs by considering those that provide high availability, profits and QoS. Although the orientation of these approaches is promising and may contribute to enhance CPs' performance, these approaches often suffer from the hazard of choosing unreliable CPs in the federation, resulting in performance degradation and loss of CPs' reputation due to Service Level Agreement (SLA) violations between the users and the federation.

Motivation. The demands for data-intensive applications and big data solutions are expected to increase in such a way that makes current Cloud Providers resources become insufficient. This promotes the need to have a platform and/or infrastructure that enable automatic scaling and reliability of recent services/applications. A Cloud Federation (CF) provides an effective platform for that by enabling CPs to upgrade their resource scaling strategies. Although several federation approaches have been proposed, they suffer from the hazard of working with untrusted providers, which results in performance degradation. Consider a CP that does not have sufficient resources to fulfill the requested VMs, and needs to outsource some of the requested VMs to other providers within the federation. If the trust issue is not considered during the federation formation, there will be several reasons that make delegated CPs unable to achieve requests as expected. These reasons include the following : inadequate maintenance causing frequent breakdowns, poor security causing compromised nodes or nodes slowed down by a DDoS (Distributed Denial-of-Service Attack) and lack of capacity for the load accepted. Also, the delegated CP might be selfish and/or malicious and refuse to share enough resources. Worse, it may even drop the requests. Such unreliable delegated CPs result in performance degradation, decreased revenues and poor users satisfaction as well as Service Level Agreement (SLA) violations.

Indeed, ''trust" has different meanings in various disciplines (e.g. psychology, politics, economy). In this paper, trust is interpreted as the degree of the belief that a delegated CP in a given federation will achieve its tasks as it should [141] [90].

To address the aforementioned problems, we propose a flexible trust-based framework for

federation formation. Our framework can be summarized as follows. We let CPs evaluate a trust value for each other by adopting two approaches : objective and subjective trust evaluations. In the former, the trust value is computed from direct observation. The CP's trust value is evaluated based on its previous interactions (i.e. experience) using Bayesian inference. In the latter, the Dempster-Shafer Theory (DST) integrated with the Bayesian approach is used when there is no previous interactions among CPs. Thereafter, we propose a federation formation algorithm that is based on the coalitional game theory [130]. The algorithm enables CPs to leave or join a given federation in such a way that enhances their trust among CPs.

Unlike similar works (e.g. [90]), we adopt a distributed mechanism in which each CP autonomously makes its own decisions. This, in turn, avoids the difficulty of finding a trusted third party. Also, it reduces instability inside the federation due to single point of failure. In summary, our work consists of the following contributions :

— Modeling and proposing a decentralized framework that considers the trustworthiness of heterogeneous CPs during the formation of cloud computing federations. More specifically, we present a systematic approach that associates the trustworthiness of CPs through the formation procedure.
— Proposing a new trust evaluation approach, based on Bayesian inference, that enables a CP to evaluate another CP's trustworthiness based on their previous interactions and experiences.
— Proposing a trust aggregation technique, based on the Dempster-Shafer Theory (DST) [16] integrated with the Bayesian approach , that enables a CP to derive a trust value in cases where there were no previous interactions and experiences.
— Devising a federation formation algorithm, based on a coalitional game theory, that allows a set of heterogenous CPs to build their federations in order to maximise their individual trust value. The proposed algorithm converges to a Nash-stable situation ; that is, no CP has a motivation to go out from its current federation and move to another federation.

The remainder of the paper is organized as follows. In Section 6.2, we discuss the related work. We present the trust model and assumptions in Section 6.3. In Section 6.4, we present the proposed federation formation framework. In Section 6.5, we present our empirical results. Finally, Section 6.6 concludes the paper.

6.2 Related Work

The concept of federations among CPs was first introduced by Rochwerger et al. [20]. Although their work shows the main materials needed to achieve federation, they did not show the architectural elements that compose multi-cloud computing environments. Buyya et al. [74] introduce the challenges and architectural elements for federations. Similarly, Celesti et al. [21] and Fazio et al. [75] present a cloud architecture that allows CPs to build a federation with each other. They consider two kinds of CPs : home and foreign. The home CPs are those that are unable to fulfill the consumers' tasks and therefore forward these jobs to the foreign CPs. Similarly, Goiri et al. [76] present a decision-based model that helps a CP decide on forming federations with public CPs in order to maximize their individual profit.

Toosi et al. [77] present multi-resource provisioning policies, that assist the CPs to increase their resource utilization and profit. Their model can terminate VMs whenever the profit of shutting them down exceeds the profit of running such VMs. Also, Van den Bossche et al. [78] present a binary integer program model that reduces the cost of outsourcing, using a mix of public and private providers. Chaisiri et al. [79] propose an optimal VM provisioning algorithm using stochastic programming that considers multi-cloud providers with the objective of maximizing their profit. Similarly, Bruneo [80] proposes a performance evaluation approach based on stochastic reward nets for federated CP. The model predicts and quantifies the cost-benefit of a strategy portfolio and the corresponding QoS experienced by clients.

A business-oriented cloud federation model for real-time applications is proposed by Xiaoyu et al. [81]. The model allows multiple heterogeneous CPs to cooperate and provide scalable infrastructure. The advantage is the business layer added to support the federations. The layer can trigger on-demand resource provisioning across multiple CPs and therefore helps maximize the clients satisfaction and business benefits [81].

Salama & Shawish [82] present a QoS-based approach for cloud federation. They use QoS metrics such as throughput and response time during the federation formation process. By considering QoS metrics, the federation helps eliminate Service Level Agreement (SLA) violations and maximise QoS targets. In [83], Mashayekhy et al. propose a hedonic coalitional game to achieve cooperation among IaaS services. Based on the federation coalition game, they design a cloud federation formation mechanism that allows CPs to form federations that maximize their profits.

A game theoretic approach for cloud federation is also proposed by Hassan et al. [84]. The study enables the dynamic resource allocation in a cloud federation. They define a price

function for a CP that gives incentives to other CPs to contribute their resources in order to form a federation. Similarly, Mihailescu & Teo [85] present a strategy-proof dynamic pricing scheme for cloud federations. In [86], Li et al. propose profit maximization strategies in cloud federations. They present a truthful auction-based mechanism for selling VMs within a federation. This enables cloud federation members to sell or buy resources in a way that maximises their profit. Also, Samaan [87] proposes an economic model for sharing resources among CPs in the federation.

Few studies have addressed trust issues in cloud federations. For example, Ngo et al. [88] present an approach for attribute-based trust establishment to be used in the multi-cloud environment. They propose an approach for trust evaluation and delegation. Messina et al. [89] suggest a trust model based on the reputation. The model allows users to properly select a suitable CP on the basis of reliability and reputation.

Hassan et al. [90] propose a trust-based hedonic game to form coalition among CPs. They enable CPs to join a coalition based on maximization of profits and minimization of penalty costs. The main limitation of their approach is that it is based on a centralized architecture and a trusted third-party is required in order to organize the coalition. In [91], Wahab et al. designed a trust-based hedonic game to the model community formation problem among multi-provider services. The main advantages of this approach lie in the (1) trust-based aggregation technique that can overcome collusion attacks in the presence of dishonest parties [91], (2) distributed trust-based coalition formation model that does not need a centralized entity, and (3) bootstrapping mechanism that assigns initial trust values for newly deployed services. The main limitations of this approach are that it considers functionally-similar services to create coalitions. Moreover, in this approach, untrusted services are considered as those that show some malicious behavior, whereas in some cases some untrusted services might be non-malicious (e.g., lack of experience). Besides, the computation of the trust values in this approach are limited to recommendations collected from different parties without considering self-experience. The trust-based cloud federation approach proposed in this paper follows the same methodology proposed in the aforementioned approach [91], while addressing the above-mentioned limitations by (1) enabling the formation of federations under a heterogeneous environment, (2) generalizing the concept untrusted agents (i.e, CPs) to cover both malicious and non-malicious CPs, and (3) calculating the trust values of CPs based on self-experience, while using a Bayesian method for this purpose.

Overall, for a multi-cloud environment, a decentralized framework that considers trustworthiness of heterogeneous CPs during the forming of federations has yet to be addressed. Thus, in this paper, we present a systematic schema that considers the trustworthiness of CPs

through the cloud federation formation process. This enhances CP's performance, revenues and clients' satisfaction.

6.3 Trust Model and Assumptions

In this section, the definition and properties of trust in the context of cloud federations are explained. Based on the definition, the trust model used to formulate trust among CPs in the federation is described. Furthermore, our methodology is also introduced. Table 6.1 summarizes the different notations that are used in this section.

Table 6.1 Notations

Symbol	Significance
F	Federation
F_k	Federation k
$\mathcal{N}$	Set of CPs
N	Number of CPs
Π	Set of federation partitions
T_i^j	The trust value of CP j with respect to the CP i.
$U_i(F_k)$	CP i's federation trust criterion of a given federation k

6.3.1 Definition of Trust

Trust can take on different meanings in different disciplines (e.g. psychology, politics, economy). In this paper, we define trust as a degree of belief that a CP, in a given federation, will achieve its tasks as expected. Based on this definition, untrusted CP are not necessary malicious. A malicious CP may accept to share other CPs with the intention of dropping other CPs' jobs and refuse to share its resources within the federation. Moreover, the CP may not be malicious yet still deficient. For example, inadequate maintenance causing frequent breakdowns, poor security causing compromised nodes or nodes slowed down by DDoS and lack of capacity for the load accepted. All of these scenarios may affect achieving tasks as expected.

6.3.2 Trust Model

The aforementioned definition is used to establish our trust model. We evaluate trust in the proposed scheme using a real number T with a continuous value between 0 and 1. In our model, trust is made up of two components : direct and indirect observations. In direct observation trust, an observer estimates the trust of a CP based on its own experience. Thus,

the trust value consists of the anticipation of a objective probability used by a CP to decide whether another CP is reliable or not. A trust value from direct observation is computed using Bayesian inference. This process is explained in detail in Section 6.4.1.

The use of direct observations requires that a trustor has had previous interactions with a CP in order to give its judgement and derive a trust value. If we only consider direct observations, evaluating the trust value of a new CP (i.e. a cloud without any previous interactions) becomes impossible. We tackle this problem by considering another way of obtaining trust. This is done by collecting judgments on this CP from other CPs that previously interacted with that CP. Therefore, the trust value becomes the anticipation of a subjective probability used by a CP to decide whether another CP is reliable or not. The Dempster-Shafer Theory (DST) [16] [138] is an adequate candidate to support this situation, in which evidences are collected from CPs that may be unreliable. A detailed explanation of this process is included in Section 6.4.2.

Based on the obtained trust values, a trust-based federation formation algorithm is presented. The algorithm is based on the coalitional game theory [130], [15], [133], where each coalition represents a given federation F.

Given the set $\mathcal{N} = \{1, 2, \ldots, N\}$ of CPs, a federation $F \subseteq \mathcal{N}$ represents an agreement among the CPs in F to delegate tasks among each other. Let $\Pi = \{F_1, ..., F_l\}$ represent the set of federation partitions. That is, for $k = \{1, 2, \ldots, l\}$, each $F_k \subseteq \mathcal{N}$ is a disjoint federation. The algorithm enables each CP $i \in \mathcal{N}$ to decide whether to join a given federation $F_k \in \Pi$ or not based on the federation trust criterion calculated by CP i. Each CP i determines the federation trust criterion $U_i(F_k)$ of any given federation $F_k \in \Pi$ by calculating the product of the trust values of all the CPs in that federation :

$$U_i(F_k) = \prod_{j \in F_k} T_i^j \tag{6.1}$$

where T_i^j denotes the trust value of CP j with respect to the CP i.

The use of the product in the definition of the federation trust criterion instead of summation, is due to the fact that the former enables a very small CP trust value to have a significant impact on the result. Therefore, its impact will not be mitigated by a higher value as in the case of summation. A detailed explanation is included in Section 6.4.3.

Based on the above discussion, the methodology of the proposed scheme is shown in Figure 6.1. In Figure 6.2, we present algorithmic steps of the proposed methodology.

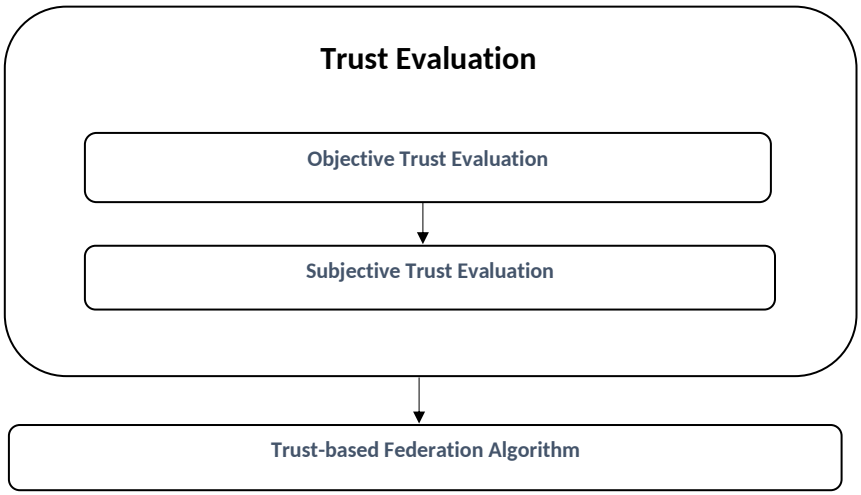


Figure 6.1 Proposed Methodology

Given the current federation partition, each CP *i* evaluates the possible shifts from its current federation as follows:

Step 1: Evaluation of the Trust Value

- Compute the trust value of each CP: if CP *i* has previous experience with CP j, evaluate CP j's trust value as in 4.1 otherwise, evaluate a trust value as in 4.2.
- Compute the overall trust value (i.e., the utility) for all existing federation include CP's current federation as in Eq. 1:

$$U_i(F_k) = \prod_{j \in F_k} T_i^j$$

Step 2: Federation Formation

- If the utility of the given federation is greater than the utility of the current federation:

  1. Leave the current and join the new one.
  2. Update the federation members according to the new shift.

Else,

Stay in the current federation.

Figure 6.2 Algorithmic steps of the proposed methodology

6.4 The Proposed Trust-based Federation Formation Framework

In this section, a trust-based federation formation framework is presented. The section is divided into the following subsections : objective trust evaluation, subjective trust evaluation and trust-based federation formation algorithm.

6.4.1 Objective Trust Evaluation : Direct Observation

In this section, the proposed objective trust evaluation approach is introduced. In this model, each CP calculates the trust value of another CP based on its experience with that provider (i.e. direct observation).

6.4.1.1 Trust computation

Based on what we presented earlier, a CP can directly evaluate the trust value of another CP based on their experience with that CP. We adopt a Bayesian inference approach to compute the trust value of a CP [14]. The Bayesian inference was chosen because it is strong approach to derive objective trust values [131]. The trust value can be enhanced if the cloud successfully performs a job as expected and it can be reduced otherwise. A CP $i \in \mathcal{N}$, uses a belief function, which computes the trust level of another CP $j \in \mathcal{N}$, that previously collaborated with CP i. Every trust value t'_j is derived from the previous or initial trust value t_j (we discuss later how to obtain the initial trust value) as follows :

$$t'_j = F(t_j; \alpha_j, \beta_j) \tag{6.2}$$

The above equation is the cumulative beta distribution function of the Beta Probability Density (BPD) function, which is defined as follows :

$$f(x; \alpha_j, \beta_j) = \frac{x^{\alpha_j - 1}(1-x)^{\beta_j - 1}}{\mathbf{Beta}(\alpha_j, \beta_j)} \tag{6.3}$$

The value of α_j and β_j are modified after the performing of each job by the CP j. β_j is increased when a CP j performs a given job as desired. Eq. (6.4) describes the modification of β_j.

$$\beta_j = \beta_j \times (1 + \mu_j) \tag{6.4}$$

where μ_j represents the weight of doing the job by CP j if it is done as desired and 0 if not.

Eq. (6.5) describes the modification of α_j.

$$\alpha_j = \alpha_j \times (1 + \nu_j) \tag{6.5}$$

where ν_j denotes the weight of doing the job by the cloud if it is done as desired and 0 if not.

The amounts of μ_j and ν_j should be carefully set by a CP i who is allocating the jobs. These values reflect the importance of the jobs assigned to the delegated CP. Note that a large amount of β_j will enhance the trust of a CP j while a small amount of α_j will reduce it.

The initial trust value t_j is calculated at the beginning through the testing period. The total number of reported jobs requests from CP j is denoted by the set $\mathcal{M}_j$. The initial trust value represents the total number of successfully performed jobs over the total number of jobs :

$$t_j = \frac{\sum_{k \in \mathcal{M}_j} r_{j,k}}{|\mathcal{M}_j|} \tag{6.6}$$

Where the parameter $r_{j,k}$ is the revealed result of the k-th job request : $r_{j,k}$ =1 indicates successful accomplishment of the k-th request. $r_{j,k} = 0$ indicates non-successful accomplishment of the k-th request.

The initial value of α and β can be obtained as follows :

$$\alpha_j = \sum_{k \in \mathcal{M}_j} (1 - r_{j,k}) \tag{6.7}$$

$$\beta_j = \sum_{k \in \mathcal{M}_j} (r_{j,k}) \tag{6.8}$$

6.4.2 Subjective Trust Evaluation : Indirect Observation

Computing the objective trust value of a CP requires some interactions to be established between CPs. In other words, an experience with a certain CP is required in order to evaluate its performance in achieving jobs. In some situations, evaluating the trust value of a CP using that method cannot be achieved without interactions between CPs. To address this problem, an DST-based aggregation technique is used [16]. DST is a mathematical theory that aggregates evidences from independent sources to come up with a degree of belief regarding a certain hypothesis [16]. DST was selected for the following main reasons : (1) unlike other aggregation approaches (e.g. Bayesian approach) that demand complete knowledge of prior probabilities, DST can handle a lack of complete knowledge (i.e. uncertainty), and (2) it has a

property to prevent collusion attacks, which occur when several malicious CPs collaborate to give misleading judgments [91]. This attack can be performed either to increase the trust value of some CPs or to decrease the trust value of other CPs. It is worth noting that a trust-based hedonic game that employs DST for feedback aggregation was first proposed in [91]. Our work uses a similar methodology for building trust. However, the main differences between our work and the aforementioned work is that our model is able to deal with both untrusted non-malicious CPs and malicious CPs. In addition, the Bayesian approach is integrated with DST in order to compute the credibility score (described bellow). It is worth also noting that DST is regarded as a useful approach in uncertain reasoning and is widely used in trust-based distributed multi-agent systems and applications (e.g., [142] [91] [143] [144] [141]).

The proposed aggregation function is defined as follows. Let $\Omega = \{T, \bar{T}, U\}$ denotes a set consisting of three hypotheses. T indicates that a certain CP j is trustworthy, $\bar{T}$ depicts that CP j is untrustworthy and U shows that j is either trustworthy or untrustworthy. A CP $a \in \mathcal{N}$ evaluates the trust value of a CP $b \in \mathcal{N}$ through combining the belief of CPs who have direct interactions with b. Each neighbor gives evidences from its own observations (i.e., direct observations) by assigning its beliefs over Ω. Each hypothesis is assigned a basic probability value (bpv) between 0 and 1, which is equal to the credibility score believed by the CP giving the judgement. We follow the work in [143] and allow bpv to be obtained from direct observations. For example, Assume that CP c believes and reports that CP b is trustworthy, then the bpv for c would be : $m_c(T) = T_{a,c}$, $m_c(\bar{T}) = 0$ and $m_c(U) = 1\text{-}T_{a,c}$, where $T_{a,c}$ is the trust value of a CP $c \in \mathcal{N}$, which is obtained from direct observations of CP a to CP c as illustrated in Section 6.4.1. On the other hand, if CP c claims that CP b is untrustworthy, then the bpv for CP c would be : $m_c(T) = 0$,$m_c(\bar{T}) = T_{a,c}$, and $m_c(U) = 1\text{-}T_{a,c}$.

The bpv for CP c can be used to define the belief function of the three hypotheses $\Omega = \{T, \bar{T}, U\}$ [16] : $belief(T) = m_c(T)$, $belief(\bar{T}) = m_c(\bar{T})$, and $belief(U) = m_c(T) + m_c(\bar{T}) + m_c(U)$. Thus, CP a can decide whether CP b is trustworthy or not based on the bpv of CP c, who has reported its judgement to CP a. To clarify these points, we will present detailed example in Section 6.4.2.1. To aggregate the different judgements from different CPs, a belief function is used. The belief function represents the total bpvs supporting a given hypothesis that belongs to the set $\Omega = \{T, \bar{T}, U\}$. For example, if we want to aggregate the belief or judgement of two CPs : $c1$ and $c2$ on $c3$, the combined belief of $c1$ and $c2$ is calculated over the same frame of discernment $\Omega = \{T, \bar{T}, U\}$ as follows [143] :

$$m_{c1}(T) \oplus m_{c2}(T) = \frac{1}{K}[m_{c1}(T)m_{c2}(T)+ \\ m_{c1}(T)m_{c2}(U) + m_{c1}(U)m_{c2}(T)] \tag{6.9}$$

$$m_{c1}(\bar{T}) \oplus m_{c2}(\bar{T}) = \\ \frac{1}{K}[m_{c1}(\bar{T})m_{c2}(\bar{T}) \\ + m_{c1}(\bar{T})m_{c2}(U) + m_{c1}(U)m_{c2}(\bar{T})] \tag{6.10}$$

$$m_{c1}(U) \oplus m_{c2}(U) = \frac{1}{K}[m_{c1}(U)m_{c2}(U)] \tag{6.11}$$

where,

$$K = m_{c1}(T) + m_{c2}(T) + m_{c1}(T) + m_{c2}(U) \\ + m_{c1}(U) + m_{c2}(U) + m_{c1}(U) + m_{c2}(T) \\ + m_{c1}(U) + m_{c2}(\bar{T}) + m_{c1}(\bar{T}) + m_{c2}(\bar{T}) \\ + m_{c1}(\bar{T}) + m_{c2}(U) \tag{6.12}$$

Here is an example. Assume the following :

$m_{c1}(T) = 0.75$ $m_{c1}(\bar{T}) = 0$ $m_{c1}(U) = 0.25$

$m_{c2}(T) = 0.6$ $m_{c2}(\bar{T}) = 0$ $m_{c2}(U) = 0.4$

by combining the above belief functions, we can obtain the result as follows :

$belief(T) = (0.75 * 0.6)+(0.75 * 0.4)+(0.6 * 0.25) = 0.9$

$belief(\bar{T}) = (0 * 0)+(0 * 0.4)+(0 * 0.25) = 0$

$belief(U) = (0.25 * 0.4) = 0.1$

This means that the trust $belief(T)$ of c3, which is obtained from indirect observation would be 0.9.

6.4.2.1 Illustrative Example

This section presents an example that shows how DST allows CP $c1$ to decide whether a given CP is trustworthy or not. This example is based on all of the information shown on Tables 6.2 and 6.3, which indicate the CPs' judgments on CP $c2$ and the credibility scores (bpv) of each cloud believed by $c1$, respectively.

Table 6.2 Clouds judgments on c2

CLOUD	**Cloud's Judgement on c2**
c3	Untrusted
c4	Untrusted
c5	Trusted

Table 6.3 Credibility scores of clouds believed by c1

CLOUD	**Credibility (bpv)**
c3	0.35
c4	0.24
c5	0.96

According to the proposed model, $c1$ decides, for each given CP, whether it is trustworthy or not. For this purpose, $c1$ aggregates all CPs judgements (or beliefs) on the given CP $c2$ as follows :

First, let's combine the beliefs of the CPs $c3$ and $c4$. We find the *bpv* for $c3$ and $c4$ as follows :

$m_{c3}(T) = 0$, $m_{c3}(\bar{T})$=0.35, $m_{c3}(U)$ = 1-0.35 = 0.65

$m_{c4}(T) = 0$, $m_{c4}(\bar{T})$=0.24, m_{c4}(U) = 1-0.24 = 0.76

Aggregate $c3$ and $c4$ judgements :

— $m_{c3}(T) \oplus m_{c4}(T) = \frac{1}{k}[m_{c3}(T)m_{c4}(T) + m_{c3}(T)m_{c4}(U) + m_{c3}(U)m_{c4}(T)]$
Where,
K =$m_{c3}(T)m_{c4}(T) + m_{c3}(T)m_{c4}(U) +$
$m_{c3}(U)m_{c4}(T) + m_{c3}(\bar{T})m_{c4}(\bar{T}) + m_{c3}(\bar{T})m_{c4}(U) +$
$mc3(U)m_{c4}(\bar{T})+m_{c3}(U)m_{c4}(U)$

K =0 * 0 + 0 * 0.76 + 0.65 * 0 + 0.35 * 0.24 +
0.35*0.76+ 0.76*0.24+0.65*0.76 = 1.02
$m_{c3}(T) \oplus m_{c4}(T) = \frac{0}{1.02}$=0

— $m_{c3}(\bar{T}) \oplus m_{c4}(\bar{T}) = \frac{1}{K}[mc3(\bar{T})mc4(\bar{T}) +$
$m_{c3}(\bar{T})m_{c4}(U) + m_{c3}(U)m_{c4}(\bar{T})]$
K= 1.02
$m_{c3}(\bar{T}) \oplus m_{c4}(\bar{T}) = \frac{0.49}{0.84} = 0.51$

— $m_{c3}(U) \oplus m_{c4}(U) = \frac{1}{K}[mc3(U) \oplus m_{c4}(U)]$
K = 1.02

$m_{c3}(U) \oplus m_{c4}(U) = \frac{(0.65*0.76)}{0.84}$= 0.59

Then, we combine the aggregated beliefs of $c3$ and $c4$'s with the beliefs of $c5$ as follows :

$m_{c34}(T) = 0$, $m_{c34}(\bar{T}) = 0.51$, $m_{c34}(U)$= 0.59

$m_{c5}(T) = 0.96$, $m_{c5}(\bar{T})$=0, $m_{c5}(U)$ =0.05

— K = $m_{c34}(T)m_{c5}(T) + m_{c34}(T)m_{c5}(U) +$
$m_{c34}(U)m_{c5}(T) + m_{c34}(T)m_{c5}(\bar{T}) +$
$m_{c34}(\bar{T})m_{c5}(U)+$
$m_{c34}(U)m_{c5}(\bar{T}) + m_{c34}(U)m_{c5}(U)$

= 0∗0.96+0∗0.05+0.59∗0.96+0.51∗0+
0.51∗0.05+0.59∗0.05+0.59∗0.05 = 0.65

— $belief(T)$= $m_{c34}(T) \oplus m_{c5}(T) = \frac{0.56}{0.65}$=**0.86**

— $belief(\bar{T})$= $m_{c34}(\bar{T}) \oplus m_{c5}(\bar{T}) = \frac{0.05}{0.65}$= 0.07

— $belief(U)$= $m_{c34}(U) \oplus m_{c5}(U) = \frac{0.02}{0.65}$= 0.03

Although both $c3$ and $c4$ judge that $c2$ is untrustworthy, $c1$'s belief in $c2$'s trustworthiness

is still high after combining $c3$ and $c4$'s belief with $c5$. The reason is that the credibility of $c5$ is higher than $c3$ and $c4$. This is considered a strong advantage of using the Dempester-Shafer Theory (DST).

6.4.3 Trust-based Federation Formation Algorithm

In this section, we model the problem of federation formation as a coalition formation cooperative game with non-transferable utility [95].

6.4.3.1 Description

Our federation algorithm is based on a hedonic game [95], [15], which is considered as a category of cooperative games [130], [15], [133]. The game assumes that each player (i.e. CP) is selfish and has its own preferences over the existing federations. The preference depends only on the federation members.

Coalitional models are usually analysed using the cooperative game theory, which tries to predict which coalitions or federations will form and the joint actions that coalitions take [130]. The services of a cooperative game will depend on which group of agents (i.e., a coalition) forms and the group of actions that the coalition takes [130] [145]. While Non-coalitional models are generally analysed using the non-cooperative game theory, which aims to predict individual agents strategies and utilities, and to find pure or mixed strategy Nash equilibrium [146].

A hedonic game was used due to the fact that finding the optimal federation structure in federation formation is NP-complete [134]. Thus, a hedonic game, which satisfies stability properties was used. By stability, we mean that none of the federation members (i.e. CPs) finds a motivation to go out from its current federation and join to another one.

To establish the model, a preference function is used. The function allows each CP to evaluate all the possible and existing federations it belongs to in order to make preferences. For any CP $i \in \mathcal{N}$, where $\mathcal{N}$ is a set of CPs, a preference relation $\succ_i$ is defined as a transitive relation, which is used over those federations that CP i can join with [95]. For any CP $i \in \mathcal{N}$, and given two federations F_1, F_2, the notation $F_1 \succ_i F_2$ indicates that CP i would like to be a member of F_1 rather than F_2.

Definition 1 (Hedonic Game). *A federation formation game is hedonic if the following two conditions are met. First, the utility of any player (e.g. CP) in a given federation depends only on the members of that federation. Second, if there is a preference over the set of possible*

federations, it is defined by a preference function.

Property. *The proposed federation game is hedonic.*

The utility of the CPs in a given federation is computed by multiplying the CP's beliefs in trustworthiness in each of the federation's members (Eq. (6.1)). Thus, the utility of CPs in a given federation is only dependent on the members of that federation. This satisfies the first condition. As for the second condition, we will define the preference function that enables CPs to have preferences over federations.

In our federation formation game, the preference function of the CPs can be defined as follows :

$$F_1 \succeq_i F_2 \Longleftrightarrow f_i(F_1) \geq f_i(F_2) \tag{6.13}$$

where F_1, $F_2 \subseteq \mathcal{N}$ are two federations containing CP i, and $f_i(.)$. There is a preference function defined as follows :

$$f_i(F_k) = U_i(F_k) = \prod_{j \in F_k} T_i^j \tag{6.14}$$

$\prod_{j \in F_k} T_i^j$ is given in Eq. (6.1) and denoted as the federation trust criterion. T_i^j is denoted as CP i beliefs in CP $j \in \mathcal{N}$. The CP i's beliefs in F_k's members is obtained either using Bayesian inference (i.e. objective trust value) if there is previous interactions from CP i and CP j as in Eq. (6.2) or by using the Dempster-Shafer Theory (i.e. subjective trust value) as in section 6.4.2. As mentioned earlier, we use the product of trust values instead of their summation (e.g., [91]) in the definition of the federation trust criterion in order to preserve the effect of small trust values on the global federations trust value. That way, the impact of a small trust value will not be mitigated by a higher one.

6.4.3.2 The Proposed Federation Formation Algorithm

The algorithm (Algorithm 5) that we propose is based on the following hedonic shift rule [95] : let $\Pi = \{F_1, ..., F_l\}$ represent the set of federation partitions. That is, for k = {1, 2, . . . , l}, each $F_k \subseteq \mathcal{N}$ is a disjoint federation. Each CP $i \in \mathcal{N}$ decides to leave its current federation $F_\Pi(i)$ to join another one $F_k \in \Pi \cup \phi$ if and only if its federation trust criterion (i.e.,

$U_i(F_k) = \prod_{j \in F_k} T_i^j$) in the new federation exceeds the one it obtains in its current federation. Leaving and joining decisions are considered selfish decisions, which means that they are made without considering their impact on the other CPs.

Algorithm 5: Trust-based Federation Formation Algorithm

Given the current federation partition $\Pi_c = \{F_1, ..., F_l\}$, each CP i evaluates possible shift from its current federation as follows :

repeat
- **foreach** $F_k \in \Pi_c \cup \phi$ **do**
  - **foreach** *CP* $j \in F_k$ **do**
    - **if** *CP* i *has previous experience with CP* j **then**
      - — calculate direct observation-based trust value of CP j.
    - **else**
      - — calculate indirect observation-based trust value of CP j.
    - **end**
  - **end**
  - calculate $U_i(F_k \cup \{i\})$ and $U_i(F_{\Pi_c}(i))$
  - **if** $U_i(F_k \cup \{i\}) > U_i(F_{\Pi_c}(i))$ **then**
    - — CP i leaves its current federation $F_{\Pi_c}(i)$ and joins the new federation.
    - — Π_c is updated : $\Pi_{c+1} = (\Pi_c \setminus \{F_{\Pi_c}(i), F_k\}) \cup \{F_{\Pi_c}(i) \setminus \{i\}, F_k \cup \{i\}\}$.
  - **else**
    - — CP i remains in the same federation so that : $\Pi_{c+1} = \Pi_c$
  - **end**
- **end**

until ε *elapses*;

In Algorithm 5, a CP i assesses all of the possible federations that it can work with. The algorithm calculates the trust value for each CP $j \in F_k$. If CP i has previous experiences with CP j, the algorithm calculates the objective trust value (i.e. direct observation trust value) as in Eq. (6.2). However, if CP i has no previous experience with CP j, an objective trust value (indirect observation trust value) is calculated as detailed in section 6.4.2. Then, the algorithm computes the federation trust criterion $U_i(F_\Pi(i))$ of its current federation F_Π(i) as in Eq. (6.13) and compares it with the federation trust criterion $U_i(F_k)$ of the federation F_k.

If the federation trust criterion of the current federation is greater than that of the federation F_k, then the CP i leaves its current federation to join F_k. Otherwise, CP i stays in its current federation. Note that after a certain fixed period of time denoted by ε, the whole steps are repeated, in order to capture the changes that may occur in the current federation partition Π_c. These modifications include changing of the CPs' trust values, the departure of existing CPs and the arrival of new CPs.

As for the computational complexity of Algorithm 5, the main complexity lies in the shifting operations, i.e. the process of finding a new federation to join, which equals $O(|\Pi_c|)$, where $|\Pi_c|$ is the number of federations in the current federation partition.

Algorithm 5 can be executed distributively. Each CP can behave independently from any other CP. We adopt the following actions based on [22] for this purpose : state recovery and atomic state update. In the former, the proposed federation algorithm assumes that each provider is able to obtain the current federation partition. We can use a state retrieval algorithm for this purpose (e.g., [135], [136]. In the later, in order to achieve correctness, the proposed federation formation algorithm assumes that the CPs current federation structure is not changed while CPs move from their current federation and join another one. For this purpose, we adopt a distributed mutual exclusion algorithm (e.g. [137]).

6.4.3.3 Analysis of the proposed Federation Algorithm

Here, the specifications of the proposed federation algorithm (Algorithm 5) are analyzed. More specifically, the three properties achieved by the proposed federation algorithm are highlighted. These properties are Nash-stability, individual-stability and convergence. It is worth noting that the methodology and the analyses below are inspired by those given in [147] [91] [22].

Theorem 2. *Algorithm 5 always converges to a final federation partition* Π_f .

Démonstration. According to the moving action from CP's current federation to any given federation as given in Algorithm 5, every move creates two distinct situations : moving to the new federation partition and to the previously visited federation partition. In the first situation, the number of moves done is finite. It is equivalent to the number of federation partitions at most cases. In the second situation, beginning from the previously visited federation partition, at certain time, the CP must either move to a new federation, and therefore leads a new partition, or it may prefer to stay in the current federation. Thus, the number of those partitions that are visited more than one time will be minimised and restricted. According to that, in all situations, Algorithm 5 will converge to a final federation structure.

□

Definition 2 (Nash-Stability). A federation structure Π is Nash-stable if no CP in Π has a motivation to go out from its current federation and join to another federation.

Theorem 3. *Any final partition* Π_f *resulting from Algorithm 5 is Nash-stable.*

Démonstration. This can be proven with a contradiction. If we Assume that the final federation partition is not Nash-stable. Thus, there exists a CP $i \in \mathcal{N}$ and a federation $F_k \in \Pi_f \cup \phi$ such that $F_k \cup i \succ_i F_{\Pi_f}(i)$ Then, CP i will move from its current federation to the new one, which makes Algorithm 5 unable to converge to a final federation partition, which contradicts with Theorem 2. □

Definition 3 (Individual-Stability). A partition Π is individually stable if no CP in Π can benefit from shifting from its current federation to another one without making the members of the latter federation worse off.

Theorem 4. *Any final partition* Π_f *resulting from Algorithm 5 is individual-stable.*

Démonstration. It has been already proven that any Nash-stable situation implies individual-stability [95]. Thus, we can conclude that Algorithm 5 converges to individual-stability.

□

Proposition 1. *Algorithm 5 confirms that for any CP* $i \in \mathcal{N}$ *that leaves its current federation* $F_\Pi(i)$ *and joins another federation* F_k*, the federation trust criterion of* $F_\Pi(i)$ *must be greater than the federation trust criterion of* F_k*.*

Démonstration. This can be proven by looking at the condition that makes a CP i moves from its current federation F_Π(i) and the federation F_k. The condition is $U_i(F_k) \succ_i U_i(F_{\Pi_c}(i))$, which ensures that CP i has a preference over the federations F_k and the current federation F_Π(i) based on the federation trust criterion.

□

6.5 Simulation Results and Analysis

To evaluate the trustworthy of the proposed approach, we test the ability the proposed method is able to enhance CPs' performance in terms of availability, response time and throughput (Same metrics used in [91]) in the presence of untrusted (malicious or not) CPs. In this section, we first explain the simulation setup used to perform our simulation and then study the performance of the proposed trust-based federation formation approach.

6.5.1 Simulation Setup

We implement our framework in a 64-bit Windows 8 environment on a host (Intel Core i7, CPU 3.60 GHz Processor and 16 GB RAM). We use Cloudsim [97] that is based on java programming language for implementing our model.

While it may be desirable to implement the proposed model using an open source cloud management system such as OpenStack, CloudStack or OpenNebula, we preferred to use Cloudsim due to the setup needed to validate the model on a large scale, with 100 CPs (see Table 6.4). Each CP is supposed to provide a computing power similar to a public cloud (e.g., Amazon). This large scale cannot be easily achieved in a local setup. Moreover, it cannot be achieved using public clouds either, because of some restrictions and regulations regarding large-scale testing [2]. This is why many research groups (e.g., [148], [149] [150]) use Cloudsim in their setup. Cloudsim allows the simulation of realistic large-scale CPs and the study of federations and their corresponding policies in terms of jobs migration, automatic scaling and reliability of services/applications [148], [149] [150].

As a simulation setup, we consider 100 CPs. We also consider 1,024 cores, 1,740 GB of memory and 225 TB of storage as the average of capacity per CP, which are the same parameters as those used in [83]. Table 4 shows the parameters used in our simulations. We set the types of VMs offered by each of the CPs to be similar to the VMs offered by Amazon EC2 [151]. The CPs offer four types of VM instances : small, medium, large and extra-large VMs as illustrated in Table 6.5.

While it is desirable to use realistic trust values for our implementation, we were unable to find a dataset that contains CPs' trust values. For this purpose, we experimentally derived trust values. The generated CPs' trust values are used through running the federation formation algorithm (Algorithm 5). We distributed the aforementioned 100 CPs into different initial federations that were built in a random fashion. We randomly set the state of each CP to be normal, untrusted non-malicious or malicious. Untrusted non-malicious CPs are those which are unable to accomplish other tasks within a certain time. On the other hand, malicious CPs are those that refuse to share their needed resources upon request. These behaviors for describing untrusted non-malicious and malicious CPs are widely used in the context of cloud computing [152] [153] [154].

To simulate an untrusted non-malicious CPs, we made the CP accept more tasks (i.e. referred as Cloudlets in Cloudsim) in such a way to largely exceed its capacity. Thus, the CP becomes unable to achieve other CPs' Cloudlets within the expected time. On the other hand, in order to simulate a malicious CP, we made the CP intentionally remove resources (i.e. VMs) given

to other CPs.

We followed the Bayesian inference explained in Section 6.4.1 to derive trust values. For each CP, a default trust value was set at 0.5 (this value is fair and indicates that the cloud status is not known yet). For each Cloudlet it receives, the new trust value is computed from the old one according to regularized incomplete beta function (Eq. 6.2). The trust value is promoted if the cloud achieves the task successfully (i.e. the task is accepted and achieved within the expected time) and it is demoted otherwise. The trust value for each cloud is calculated after performing 100 Cloudlets.

After obtaining trust values, we applied the proposed federation formation algorithm (Algorithm 5) on the considered providers. We compared our model with two state-of-the-art benchmarks : (1) Grand federation [74] and (2) QoS-based cloud federation [82]. In the former, the federation is formed among all CPs in order to enhance the utility. In the latter, the federation formation considers QoS metrics such as throughput and response time during the federation formation process. Thus, only CPs that have QoS metrics within the average and standard deviation are considered in the federation. We chose to compare our work with these two approaches as they are very similar to our model in their objective, which enhances the performance of individual CPs. Thus, other business-oriented federation approaches (e.g. [90] [83] [86] [81]), whose objective is to increase CPs' profit will not be considered in our comparisons.

Table 6.4 Parameters

Parameters	**Value(s)**
Number of Cloud Providers	100
Number of VM types	4
Number of cores	[512, 1536]
Memory (GB)	[870, 2610]
Storage (TB)	[112, 338]

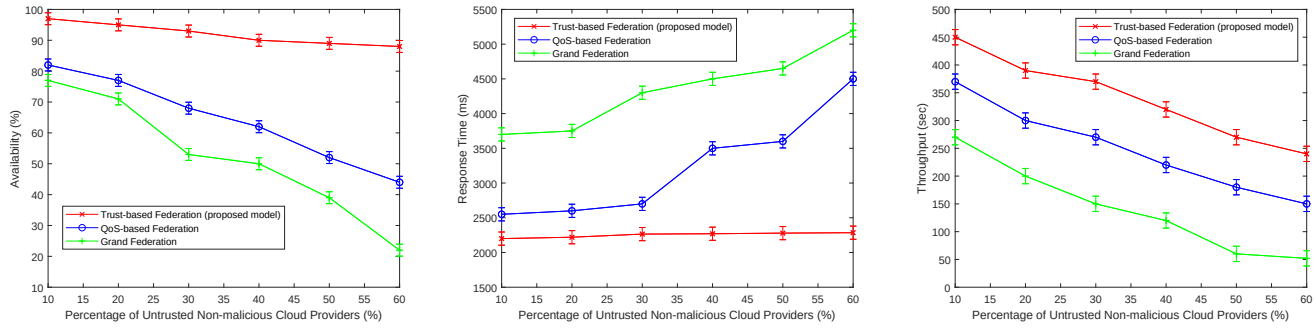
Table 6.5 The Characteristics of Available VM Instances

	Small VM	**Med. VM**	**L VM**	**XL VM**
1.6 GHZ CPU	1	2	4	8
GB Mem.	1.7	3.75	7.5	15
TB storage	0.22	0.48	0.98	1.99

6.5.2 Simulation Results

In this section, the performance of the generated federations is tested. A total of 10 000 requests (i.e., Cloudlets) is assigned for every federation, which is realistic to a large degree.

Figures 6.3a, 6.3b, and 6.3c illustrate the performance of our framework with respect to only the number of untrusted non-malicious CPs at this stage. We investigate how effective are the formed federations in terms of availability. More specifically, Figure 6.3a shows the efficiency of the formed federation in terms of availability. Availability depicts the time period in which a federation of CPs is ready for use and is obtained by dividing the number of performed requests by the total number of received requests. The figure reveals that our model outperforms both the Grand and QoS-based federation formation approaches, whose performance begins to decrease drastically. This is due to the fact that the proposed trust-based federation formation algorithm (Algorithm 5) compares all of the possible federations and builds preferences among them based on the federation trust criterion (Eq. (6.13)), which, in turn, reduces the percentage of untrusted non-malicious CPs in the federated cloud. In other words, it reduces the number of CPs that keep receiving requests that largely exceed their capacity. Figure 6.3b depicts the performance of the produced federations in terms of



(a) (b) (c)

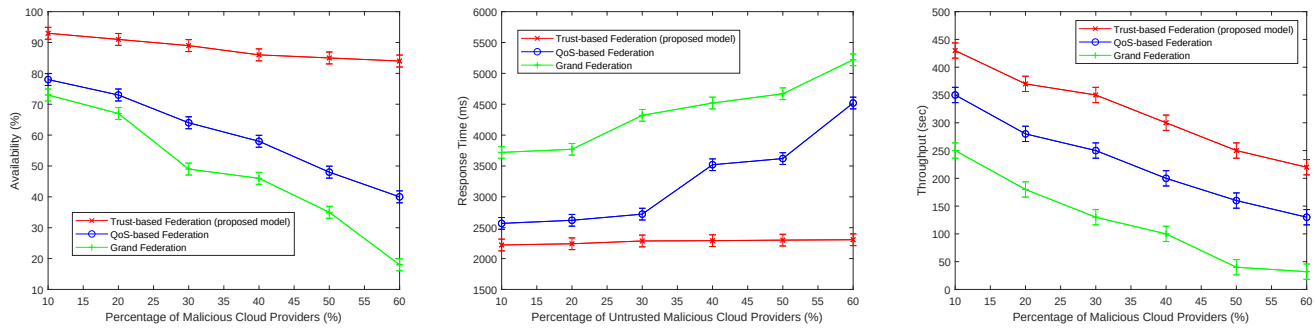
Figure 6.3 The proposed trust-based model improves the availability, response time, and throughput, compared to the Grand and QoS-based federations, in the presence of untrusted non-malicious CPs.

response time. The response time represents the time span between the submission and the response of the request, which includes both the execution and the waiting times. Figure 6.3b reveals that our model yields much less response time compared to the Grand and QoS-based models in the presence of untrusted non-malicious CPs. This is also due to the fact that our trust-based model reduces the percentage of untrusted non-malicious CPs in the federation.

Figure 6.3c, illustrates how effective are the formed federations in terms of throughput.

Throughput describes the number of requests that a federation can handle in a given time. In our simulations, throughput was measured per second. Figure 6.3c reveals that our trust-based model yields much higher throughput compared to the other two models in the presence of untrusted non-malicious CPs for the aforementioned reasons.

Next, Figures 6.4a, 6.4b, and 6.4c indicate the performance of our framework with respect to the number of malicious CPs. We analyse how effective are the formed federations in terms of availability, response time and throughput. More specifically, Figure 6.4a shows the efficiency of the formed federation in terms of availability. The figure reveals that our model outperforms both the Grand and QoS-based models whose performance begins to decrease largely. This is due to the fact that the proposed trust-based federation formation algorithm (Algorithm 5) compares all the possible federations and builds preferences over them based on the federation trust criterion (Eq. (6.13)), which in turn reduces the percentage of malicious CPs in the federated CPs. In other words, it reduces the number of CPs that refuse to share their requested resources. Figure 6.4b displays the performance of the produced federations



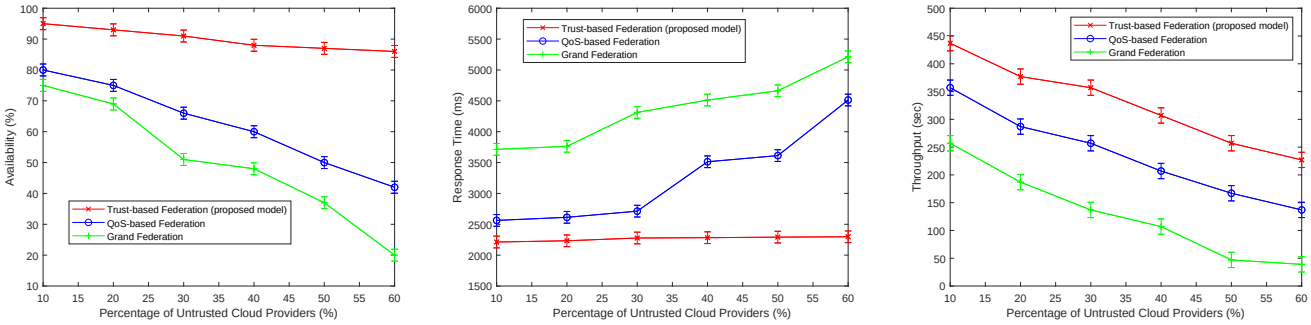
(a) (b) (c)

Figure 6.4 The proposed trust-based model improves the availability, response time, and throughput, compared to the Grand and QoS-based federations, in the presence of malicious CPs.

in terms of response time. The figure reveals that our model yields lower response time compared to the Grand and QoS-based models in the presence of malicious CPs. This is also due to the fact that our trust-based model reduces the percentage of malicious CPs in the federation according to the preference function in Algorithm 5. Also in Figure 6.4c, we see the efficiency of the formed federations in terms of throughput, which reveals that our trust-based model yields much higher throughput compared to the other two models in the presence of malicious CPs.

Also, in Figures 6.5a, 6.5b, and 6.5c we study the performance of our framework with respect to the combination of both untrusted non-malicious and malicious CPs. We analyse

how effective are the formed federations in terms of three performance metrics : availability, response time and throughput. Figure 6.5a shows the efficiency of the formed federation in terms of availability. It reveals that our approach outperforms both the Grand and QoS-based models whose performance begins to decrease largely. This is also due to the fact that our model (through Algorithm 5) compares all the federations in the federation partition and builds preferences over them based on the federation trust criterion, which in turn reduces the percentage of both untrusted non-malicious CPs and malicious CPs in the federated CPs. In other words, it reduces the number of CPs that 1) keep receiving requests that extremely exceed their capacity (untrusted non-malicious CPs) and 2) refuse to share their requested resources (malicious CPs). Figure 6.5b shows the performance of the produced federations in



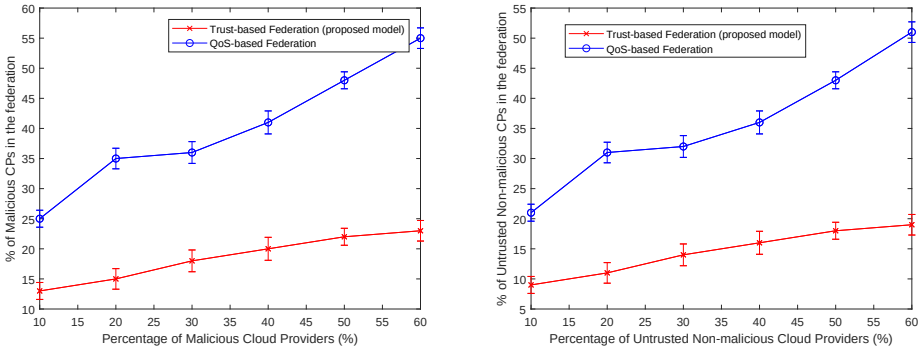
(a) (b) (c)

Figure 6.5 The proposed trust-based model improves the availability, response time, and throughput, compared to the Grand and QoS-based federations, in the presence of a combination of both untrusted non-malicious and malicious providers.

terms of response time. It reveals that our model yields lower response time compared to both Grand and QoS-based models in the presence of both untrusted non-malicious and malicious CPs. This is also due to the fact that our model reduces the percentage of both untrusted non-malicious and malicious CPs in the federation (Algorithm 5). Also in Figure 6.5c, we can see the efficiency of the formed federations in terms of throughput, which reveals that our trust-based model yields much higher throughput compared to Grand and QoS-based models in the presence of both untrusted non-malicious CPs and malicious CPs.

We now carefully investigate the reasons why our model outperforms the other studied models. More specifically, we study the percentage of malicious and untrusted non-malicious CPs that exist in the final federations. Figure 6.6b shows the percentage of malicious CPs that exist in the final federations structure with respect to the percentage of malicious CPs that existed in the initial federation. In other words, our goal is to study how effective is each of the compared models in avoiding the malicious CPs during federation's formation. We

exclude the Grand federation approach in this study as we will end up with a single Grand federation in which all CPs are members. Figure 6.6b shows that the percentage of malicious



(a) (b)

Figure 6.6 The proposed trust-based model reduces the number of untrusted non-malicious and malicious CPs.

CPs in the final federations keeps increasing in the QoS-based, and our trust-based federation formation models (i.e. our model) with the increase in their percentage in the initial partition. However, our model is more resilient to that increase and is able to minimise the percentage of untrusted non-malicious CPs up to 35% compared to the QoS-based federation approach. The reason is that our model takes into consideration the trust relationships among CPs and excludes malicious CPs during federation formation. Figure 6.6a depicts the efficiency of each of the compared models in avoiding the untrusted non-malicious CPs during federation's formation. Figure 6.6a shows that the percentage of untrusted non-malicious CPs in the final federations also keeps increasing in the QoS-based, and our trust-based federation formation models (i.e. our model) with the increase in their percentage in the initial partition. However, our model is more resilient to that increase and is able to reduce the percentage of malicious CPs up to 27% compared to the other models.

6.6 Conclusion

In this paper, we proposed a decentralised framework that considers trustworthiness of heterogeneous CPs during the formation of cloud computing federations. We proposed two approaches for evaluating the CPs' trust values : objective and subjective trust evaluations. In the former, the trust value is evaluated based on the history of interactions (i.e. experience) using Bayesian inference. In the latter, we used the Dempster-Shafer Theory (DST) of evidence integrated with the Bayesian inference to evaluate the trust value in the absence

of previous interactions. Thereafter, we devised a federation formation algorithm, based on the coalitional game theory, that allows a set of CPs to cooperatively set up their federations in order to maximise the trust of the formed federations. We have shown that the proposed federation formation algorithm converges to a Nash-stable situation, i.e. no CP has a motivation to go out from its current federation and move to another federation. Our experimental results show that the proposed federation formation algorithm minimizes the number of malicious and untrusted CPs in the federation up to 35% compared to two state-of-the-art federation approaches. Moreover, the performance of the formed federations in terms of availability, response time and throughput is improved.

CHAPTER 7 ARTICLE 4 : MULTI-CLOUD COOPERATIVE INTRUSION DETECTION SYSTEM : TRUST AND FAIRNESS ASSURANCE

Adel Abusitta, Martine Bellaïche and Michel Dagenais
Annals of Telecommunications (Submitted).

Abstract

The sophistication of the recent cloud computing systems has made them more vulnerable to intelligent cyber attacks. Moreover, it is becoming very difficult for a single intrusion detection system (IDS) to detect all existing attacks, due to limited knowledge about such attacks' patterns and implications. Recent works in cloud security have shown that a cooperation among cloud-based IDSs can bring higher detection accuracy compared to one traditional IDS. Such a cooperation allows a cloud-based IDS to consult other IDSs about suspicious intrusions and to hence increase the decision accuracy. However, there are two main challenges associated with the existing cooperative IDSs, which are related to trust and fairness assurance. To tackle these challenges, we propose in this paper a cooperative cloud-based IDS framework that 1) enables IDSs to distributively form trustworthy IDSs communities by advancing a trust-based hedonic coalitional game, which allows IDSs to increase their individual detection accuracy in the presence of untrusted IDSs and 2) formulates a fairness assurance mechanism as a Stackelberg game between the well-behaving IDSs and the selfish ones that frequently send consultation requests to other IDSs, and at the same do not answer other IDSs' consultation requests. Experimental results show the effectiveness of the proposed approach in terms of enhancing the accuracy of detection and achieving the fairness among IDSs in terms of benefits obtained through cooperation.

7.1 Introduction

The virtualization layer added to cloud computing systems has made them more vulnerable to security threats compared to traditional computing systems. Cyber attacks are also becoming more sophisticated and harder to detect. Therefore, it is becoming increasingly difficult for a traditional single intrusion detection system (IDS) to detect all attacks, due to limited knowledge about attacks. A collaboration among IDSs has proven its efficiently in terms of the accuracy in detecting new and sophisticated attacks [63] [64] [65]. Through collaboration, IDSs in different regions, and possibly, belonging to different Cloud Providers (CPs) can

cooperate in such a way to utilize the expertise of each other to cover unknown threat patterns. This can be done by enabling IDSs to consult each other about suspicious behavior, where the received feedback can be then used to decide whether to raise an alarm or not. Recent works [11] [23] show that by collecting feedback from other IDSs, the detection rate can be enhanced up to 60%.

Several approaches have been proposed to model the cooperation among cloud-based IDSs (e.g. [63] [64] [65] [66] [128] [129]). However, these approaches work under the assumption that all IDSs are trustable, which makes their collaboration systems vulnerable to untrusted (malicious or not) insiders. In this paper, we propose a trust-based framework for cooperative IDS in a multi-cloud environment. The proposed model enables an IDS to evaluate other IDSs' trustworthiness by modeling its personal experience using Bayesian inference. After obtaining IDSs' trust values, a community formation algorithm is proposed. This algorithm is based on the coalitional game theory [130]. The algorithm enables IDSs to join and/or leave a given community so as to enhance their chances of working with trusted IDSs. The proposed algorithm enables each CP to discover those CPs that have trusted IDSs, and list them on its whitelist. Figure 7.1 shows the architecture of the proposed cooperative IDS. In Figure 7.1, we can see that CP4 has a whitelist of several CPs that employ trusted IDSs. As shown in Figure 7.1, CP4 sends a consultation request to its whitelist for intrusion diagnosis. The IDSs in the whitelist send their feedback to the CP4'IDS, which then uses the Dempster-Shafer Theory (DST) [16] for feedback aggregation. DST allows us to handle the lack of complete information, and to also prevent collusion attacks, which occur when several IDSs collaborate to give misleading judgments.

Although achieving the trustworthiness of the cooperative detection process can enhance the community's members performance, the motivation of the cloud-based IDSs to participate in the detection process needs further investigation. An IDS can be trustworthy but selfish at the same time. The selfish property means that the IDS frequently sends consultation requests to other IDSs and at the same time does not answer other IDSs' consultation requests with the aim of saving its own resources. Therefore, the fairness property of the cooperation model needs to be incorporated in the framework so as to achieve an incentive compatible cooperation model. We propose a fairness assurance mechanism in order to prevent the community from *selfish* IDSs. This is important to encourage IDSs to participate in the cooperation process. The proposed fairness assurance mechanism is modeled as a Stackelberg game [17] in which each *well-behaving* IDS plays as the *leader* of the game and the *selfish* IDS is the *follower*. The strategy of the *selfish* IDSs is to maximise their consultation rates and at the same time minimise their response rates. Knowing this strategy, the strategy of the *well-behaving* IDSs is to choose the optimal response rates that is fairly compatible with

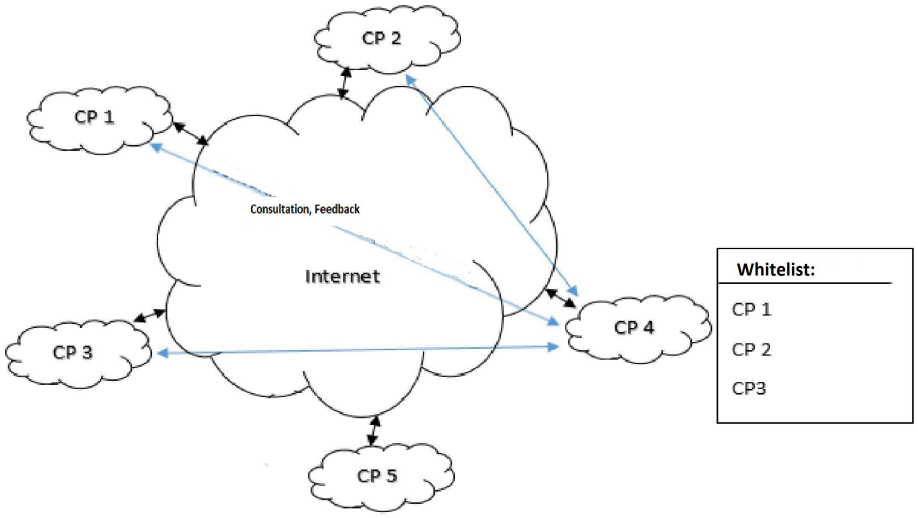


Figure 7.1 Architecture of the proposed cooperative IDS

their consultation rates. We solved the optimization problem using the backward induction reasoning [98] through finding at the beginning the best response of the *selfish* IDS to the *well-behaving* IDS's consultation rate strategy and then merging this information into the *well-behaving* IDS's optimization problem. The outcome of the game is the optimal response rate for the *well-behaving* IDS.

The major contributions of this paper are as follows : 1) Modeling and proposing a framework that enables cloud-based IDSs to distributively form trustworthy IDS communities. 2) Proposing a new trust evaluation technique, based on Bayesian inference, that allows a cloud-based IDS to evaluate another IDS's trustworthiness based on its personal experiences. 3) Devising an algorithm, based on the cooperative game theory, that allows a set of cloud-based IDSs to cooperatively set up their community in such a way to increase their individual detection accuracy in the presence of untrusted IDSs. The proposed algorithm converges to a Nash-stable situation ; that is, no IDS has a motivation to leave its current community and move to another one. 4) Proposing a fairness assurance mechanism that prevents *selfish* IDSs in the communities.

This work is an extension of the previous work proposed in [12], where we present a trust-based game theoretical approach in cloud-based IDSs. This paper extends our previous work by (1) presenting more analysis and a mathematical description about the proposed turst-based community formation model ; (2) formulating a fairness assurance mechanism between the *well-behaving* IDSs and the *selfish* ones that frequently send consultation requests to

other IDSs, and at the same do not answer other IDSs' consultation requests ; and (3) presenting more information with a concrete example about the proposed feedback aggregation approach, which is used inside the formed communities in order to make decisions about suspicious intrusions. Moreover, we add more results and discussion about the proposed approach.

The rest of this paper is organized as follows. In Section 7.2, we discuss the related work. We present the trust-based cooperative intrusion detection system in Section 7.3. In Section 7.4, we present the fairness assurance mechanism. In section 7.5, we present our empirical results to show the effectiveness of the proposed approach. Finally, Section 7.6 concludes the paper.

7.2 Background and Related Work

Cloud-based IDSs can be classified into two types ; signature-based and anomaly-based [1]. The former compares suspicious behavior with known attack patterns. In order to make signature-based systems effective, the signature database should be updated frequently. On the other hand, anomaly-based IDSs raise alarms when unusual and/or unexpected activities are detected. Anomaly-based IDSs are effective in detecting unknown attacks. Moreover, they do not need a database of known attacks. However, the shortcoming of using anomaly-based detection lies in the relative high false positive rate compared to the signature-based technique [11]. IDSs may adopt both techniques to have an improved detection accuracy. However, the detection accuracy is limited by the amount of knowledge IDSs' have (e.g., their security vendors have). Recent research has shown that the collaborative detection can enhance the detection rate up to 60% [11] [23]. In this section, we present the state-of-the-art of the cloud-based cooperative IDSs.

Cooperative IDSs in the context of cloud computing have been proposed in many earlier works . For instance, Lo et al. [62] propose a cooperative detection method in the virtualized cloud environment. Their method allow alerts to be exchanged among different nodes (i.e., hosts) whenever an attack gets detected. For this purpose, they adopt a rule-based technique to identify TCP SYN attacks by fetching the threshold for rule patterns during the initial rule establishment phase. The advantage of this method is that it is able to balance the detection overhead among nodes. Also Teng et al. [61] proposed a method that aggregates two types of detectors : a feature detector and a statistical detector. The former uses SNORT to separate events based on network protocols (e.g., TCP). The later cooperates with the feature detector by using data packets from it to decide whether an event is an attack or not. If the rate (i.e., the rate of packets) obtained is grater than the predefined threshold, then this situation will be considered as an attack.

Man and Huh [63] and Singh et al. [64] propose a cooperative IDS between cloud computing regions. Their approaches enable exchanging alerts from multiple elementary detectors. In addition, they enable the share of information between interconnected clouds. Also, Ghribi [65] proposed a middleware IDS. The approach allows a cooperation between three layers : Hybervisor-based IDS, Network-based IDS and VM-based IDS. If an attack is found in a layer, the attack cannot be executed in the other layers. Chiba et al. [66] propose a cooperative network-based cooperative intrusion detection system to detect network attacks in the cloud. This can be performed through traffic monitoring while maintaining performance and service/application quality.

The main shortcoming of the above works is that they consider that all cloud-based IDSs are trustable, which lets their collaboration systems more vulnerable to untrusted and/or malicious insiders. The goal of this article is to present a systematic approach to establish a cloud-based cooperative IDS that uses trust assessment mechanisms and enables trustworthy decisions aggregation . We aim to allow our approach to work in the presence of untrusted and/or malicious IDSs .

In a multi-cloud environment, Dermott et al. [67] propose a cooperative intrusion detection in a federated virtualized cloud. They adopt the Dempster-Shafer theory of evidence to gather the beliefs given by the watching entities. The gathered beliefs are used to let the final decision regarding a possible attack. The main shortcoming of this approach is that it is a centralized-based architecture, whereby a trusted third-party should collect and manage feedbacks.

Cooperative IDSs in non-cloud environments where also proposed recently, in [68] [69] [5] [6] [7] [8] [9] [10]. They have the shortcoming as the above mentioned works, since they assume that all IDSs are trustable, which makes their collaboration system vulnerable to malicious insiders.

A trust-based cooperative IDS has been proposed in a non-cloud environment in [11]. They propose a trust-based collaborative decision framework. Through collaboration, a native IDS can identify new attacks that may be known to other IDSs. The work evaluate how to utilize different diagnosis coming from different IDSs. They propose a system architecture for a collaborative IDS where trustworthy feedback aggregation is a key component. Similarly, Zhu et al. [70] [71] propose an incentive-based communication protocol, which gives IDS nodes incentives to share their feedback, and thus to prevent untrusted behaviors. The main shortcoming of these works is that they consult many IDSs in order to get a feedback. This, in turn, causes extra overhead, through consulting needlessly some IDSs (i.e., untrusted IDSs). This is unlike our approach, where we adopt a trust-based coalitional game approach, in

order to construct a set of trusted IDSs and thus reduce the rate of consultation requests while ensuring a higher detection accuracy.

A fairness assurance mechanism in a cooperative IDS also has been proposed in [72] and [73]. They create a rule dissemination protocol based on a decentralized two-level optimization framework, which determines the information propagation rates to each IDS. For this purpose, they adopt a Bayesian learning approach for the IDS to find the compatibility ratio of other IDSs based on the historical interactions gathered by each IDS. The main limitation of their approach is that it is limited to signature-based cooperative IDSs, where the fairness is measured in terms of the ability to distribute rules fairly among IDSs.

A trust-based hedonic game to form a community among CPs is presented in [90]. The approach allows CPs to dynamically join a community based on the maximization of profits and minimization of penalty costs. The main limitation of this work is that it is based on a centralized architecture, whereby a trusted third-party called broker [155] is responsible for formatting the community. A decentralized trust-based hedonic game has also been proposed by Abdel Wahab et. al. [91] in order to model communities among functionally-similar Multi-cloud services. To create a community, the services have to work as one entity, called community. The main advantages of this approach are 1) proposing a trust-based aggregation technique that is resilient to collusion attacks [91], 2) proposing a trust-based community formation model that does not rely on a centralized architecture, and 3) proposing a bootstrapping algorithm in order to give new trust values for new services. However, the main limitation of this approach is that it works only under homogeneous environment for forming communities. Also, untrusted parties are those considered to have a malicious behavior, where in some cases untrusted agents can also be non-malicious (e.g., lack of experience). Another limitation of that approach is that they let a trust value to be computed based on recommendations received from other agents, where a Dempster-Shafer approach has been proposed for feedback aggregation. This prevent trust values to be computed based on an agent's self experience. In this paper, the proposed trust-based community formation for cooperative multi-cloud IDSs uses the same methodology proposed in the aforementioned approach [91]. However, we have addressed the above mentioned limitation by 1) allowing the community to be established under a heterogeneous environment, 2) generalizing the concept untrusted agents (i.e, IDSs) in order to represent both malicious and non malicious IDSs and 3) allowing trust values of an agent to be computed based on self-experience, where we propose a Bayesian approach for this purpose.

Overall, for a multi-cloud environment, a decentralized framework that considers trustworthiness of IDSs, and guarantees the fairness among them during the cooperation, is yet to be

addressed. Therefore, in this paper, we present a trust-based cooperative IDS in a multi cloud environment in order to enhance the detection accuracy compared to the existing cooperative and non-cooperative IDSs. Moreover, the proposed solution makes the collaboration solution to be fair, incentive compatible, and efficient.

7.3 The Proposed Trust-based Cooperative IDS

In this section, we present a framework for the proposed trust-based cooperative IDS in a multi-cloud environment. The framework is divided into the following components as shown in Figure 7.2 : trust evaluation, trust-based community formation algorithm and trust aggregation.

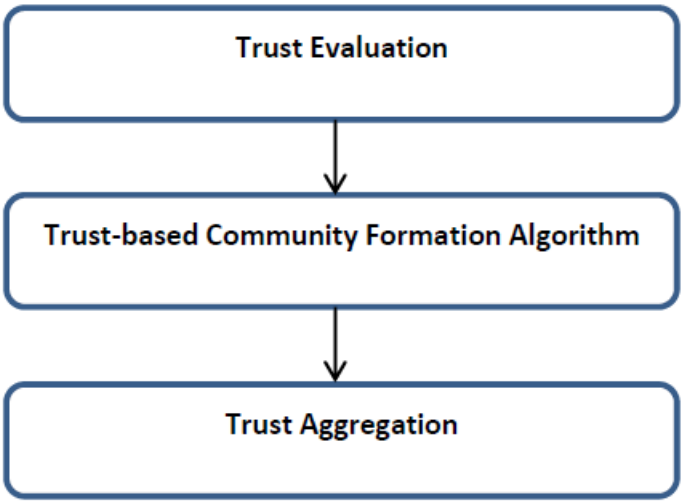


Figure 7.2 The Trust-based Cooperative IDS (Methodology)

7.3.1 Trust Evaluation

A cloud-based IDS can estimate the trust value of another IDS based on its experience with that IDS. We use a Bayesian inference approach to compute the trust value of an IDS [14], which is a well-known approach to derive and evaluate trust values [131]. When the cloud-based IDS asks another IDS aboud a suspicious intrusion, the received feedback and the revealed result (i.e., attack or not) are used to calculate the trust value of the consulted IDS. The trust value can be enhanced if the IDS successfully diagnosed the consultation request about a suspicious intrusion and it can be reduced otherwise. The trust value here represents

the accuracy of the IDS diagnosing suspicious intrusions. An IDS $i \in \mathcal{N}$, where $\mathcal{N}$ is a set of IDSs, is supported with a belief function, which computes the trust level of another IDS $j \in \mathcal{N}$. Every trust value t'_j is calculated from the previous initial or default trust value t_j (we discuss later how to find the initial trust value) as follows :

$$t'_j = F(t_j; \alpha_j, \beta_j) \tag{7.1}$$

The above equation is the cumulative beta distribution function of the Beta Probability Density (BPD) function, which is defined as follows :

$$f(x; \alpha_j, \beta_j) = \frac{x^{\alpha_j - 1}(1-x)^{\beta_j - 1}}{\mathbf{Beta}(\alpha_j, \beta_j)} \tag{7.2}$$

The value of α_j and β_j are modified after receiving the feedback from an IDS j. β_j is increased when the IDS j successfully diagnoses the consultation request. The following function describes the update of β_j.

$$\beta_j = \beta_j \times (1 + \mu_j) \tag{7.3}$$

where μ_j represents the weight of the diagnosed consultation request if it is successful and 0 if not.

The equation below describes the update of α_j.

$$\alpha_j = \alpha_j \times (1 + \nu_j) \tag{7.4}$$

where ν_j denotes the weight of the diagnosed consultation request if it is unsuccessful and 0 if not.

The values for μ_j and ν_j should be selected carefully by an IDS i who is receiving feedback about a suspicious intrusion from peers. These values reflect the detection difficulty degree of the suspicious intrusion. A higher amount of β_j will enhance the trust of an IDS j while a higher value of α_j will reduce it.

The initial trust value t_j is calculated at the beginning through the testing period. Let the total reported diagnosis data from peer IDS j be denoted by the set $\mathcal{M}_j$. The initial value of the trust is the total number of consultation requests that have been successfully diagnosed divided by the total number of consultation requests :

$$t_j = \frac{\sum_{k \in \mathcal{M}_j} r_{j,k}}{|\mathcal{M}_j|} \tag{7.5}$$

Where the parameter $r_{j,k}$ is the revealed result of the k-th diagnosis request : $r_{j,k} = 1$ indicates successful diagnosis of the k-th request. Conversely, $r_{j,k} = 0$ indicates non-successful diagnosis of the k-th request.

The initial value of α and β can be obtained as follows :

$$\alpha_j = \sum_{k \in \mathcal{M}_j} (1 - r_{j,k}) \tag{7.6}$$

$$\beta_j = \sum_{k \in \mathcal{M}_j} (r_{j,k}) \tag{7.7}$$

7.3.2 A Trust-based Community Formation

In this section, we model the problem of cloud-based cooperative IDS as a coalitional game [95].

7.3.2.1 Description

The proposed community formation algorithm is based on a hedonic game [95], [15] [91] [103], which is considered as a category of coalition formation games [130], [15], [133]. The game assumes that each player (i.e. IDS) is selfish and has its own preferences over the existing communities. A hedonic game is selected due to the fact that finding the optimal community structure, in community formations, is NP-complete [134]. Therefore, a hedonic game, which satisfies the stability features is used. Stability indicates that none of the community members (i.e. IDSs) finds a motivation to leave its current community and join another one.

In order to construct the model, we need to define a preference function. The function allows each IDS to evaluate all the possible and existing communities it belongs to in order to define preferences. For any IDS $i \in \mathcal{N}$, where $\mathcal{N}$ is a set of IDSs, a preference relation $\succ_i$ is defined over the set of all communities that IDS i can form [95]. For any IDS $i \in \mathcal{N}$, and given two communities C_1, C_2, the notation $C_1 \succ_i C_2$ indicates that IDS i prefers being a member of C_1 rather than C_2.

In our community formation game, the preference function of the IDSs is defined as follows :

$$C_1 \succeq_i C_2 \Longleftrightarrow f_i(C_1) \geq f_i(C_2) \tag{7.8}$$

where C_1, $C_2 \subseteq \mathcal{N}$ are two communities containing IDS i, and $f_i(.)$ is a preference function defined as follows :

$$f_i(C_k) = U_i(C_k) = \prod_{j \in C_k} T_i^j \tag{7.9}$$

$\prod_{j \in C_k} T_i^j$ is denoted as the community trust criterion. T_i^j is denoted as IDS i beliefs in IDS $j \in \mathcal{N}$. IDS i's beliefs in C_k's members is obtained using Bayesian inference as in (1). We use the product of IDSs trust values instead of their summation in the definition of the community trust criterion in order to conserve the effect of small trust values on the global communities trust value. That way, the impact of a small trust value will not be mitigated by a higher one.

7.3.2.2 The Proposed Community Formation Algorithm

The proposed community formation algorithm (Algorithm 6) is based on the hedonic shift rule [95] : let $\Pi = \{C_1, ..., C_l\}$ represent the set of community partitions. That is, for k = {1, 2, . . . , l}, each $C_k \subseteq \mathcal{N}$ is a disjoint community. Each IDS $i \in \mathcal{N}$ decides to leave its current community $C_\Pi(i)$ to join another one $C_k \in \Pi \cup \phi$ if and only if its community trust criterion (i.e., $U_i(C_k) = \prod_{j \in C_k} T_i^j$) in the new community is greater than the one it obtains in its current community. Leaving and joining decisions are considered selfish decisions. This means that they are made without considering their impact on the other IDSs.

In the trust-based community formation algorithm (Algorithm 6), an IDS i evaluates all of the possible communities it can join or form, beginning by leaving its current community C_Π(i) to join another already existing community C_k. The algorithm computes the trust value for each IDS $j \in C_k$ as in (1). Then, the algorithm determines the community trust criterion $U_i(C_\Pi(i))$ of its current community C_Π(i) as in (9) and compares it with the community trust criterion $U_i(C_k)$ of the community C_k. If the community trust criterion of the current community is greater than that of the community C_k, then the IDS i leaves its current community to join C_k. Otherwise, IDS i remains in its current community. One should note that, after a certain fixed period of time ε, the whole sequence is repeated, in order to capture the changes that may happen in the current community partition Π_c. These changes include changes in the

Algorithm 6: Trust-based Community Formation Algorithm

Given the current community partition $\Pi_c = \{C_1, ..., C_l\}$, each IDS i estimates possible shift from its current community as follows :

repeat
  foreach $C_k \in \Pi_c \cup \phi$ **do**
    foreach *IDS* $j \in C_k$ **do**
      — determine the trust value of IDS j.
    end
  end
  determine $U_i(C_k \cup \{i\})$ and $U_i(C_{\Pi_c}(i))$
  if $U_i(C_k \cup \{i\})$ *is grater than* $U_i(C_{\Pi_c}(i))$ **then**
    — IDS i leaves its current community $C_{\Pi_c}(i)$ and joins the new community.
    — Community partition Π_c is updated :
    $\Pi_{c+1} = (\Pi_c \setminus \{C_{\Pi_c}(i), C_k\})$
    $\cup \{C_{\Pi_c}(i) \setminus \{i\}, C_k \cup \{i\}\}$.
  else
    — IDS i stays in the same community :
    $\Pi_{c+1} = \Pi_c$
  end
until ε *elapses*;

trust values of the IDSs, and the departure/arrival of existing/new IDSs.

The main complexity of Algorithm 6 lies in the shifting steps, i.e. the process of finding a new community to join, which equals $O(|\Pi_c|)$, where $|\Pi_c|$ is the number of communities in the current community partition.

Algorithm 6 can be executed distributively. Each IDS can behave independently from any other IDS. We adopt the following actions based on [22] for this purpose : state recovery and atomic state update. In the former, the proposed community algorithm assumes that each provider is able to obtain the current community partition. We can use state retrieval algorithm for this purpose (e.g., [135], [136]. In the later, in order to achieve correctness, the proposed community formation algorithm assumes that the CPs current community structure is not changed while IDSs move from their current community and join another one. For this purpose, we adopt a distributed mutual exclusion algorithm (e.g. [137]).

7.3.2.3 Analysis of the proposed Trust-based community formation Algorithm

We analyse here the specifications of the proposed trust-based community formation algorithm (Algorithm 6). More specifically, the three properties achieved by the proposed coalition formation algorithm are highlighted. These properties are Nash-stability, individual-stability and convergence. It is worth noting that the methodology and the analyses presented here are inspired by those given in [147] [91] [22].

Theorem 5. *Algorithm 6 always converges to a final partition* Π_f .

Démonstration. According to the movement action from the IDS's current community to any given federation as given in Algorithm 6, every move creates two distinct situations : moving to the new federation partition and to the previously visited community partition. In the first situation, the number of moves is finite. It is equivalent to the number of community partitions in most cases. In the second situation, beginning from the previously visited community partition, at certain time, the IDS must either move to a new community, and therefore leads a new partition, or it may prefer to stay in the current community. Thus, the number of those partitions that are visited more than one time will be minimised and restricted. According to that, in all situations, Algorithm 6 will converge to a final community structure.

□

Definition 4 (Nash-Stability). A community structure Π is Nash-stable if no IDS in Π has a motivation to leave its current community to join another community.

Theorem 6. *Any final partition* Π_f *resulting from Algorithm 6 is Nash-stable.*

Démonstration. This can be proven with a contradiction. If we assume that the final partition Π_f is not Nash-stable. Thus, there exists an IDS $i \in \mathcal{N}$ and a community $C_k \in \Pi_f \cup \phi$ such that $C_k \cup i \succ_i C_{\Pi_f}(i)$ Then, IDS i will move from its current federation to the new one, which makes Algorithm 6 unable to converge to a final community partition, which contradicts Theorem 5. □

Definition 5 (Individual-Stability). A partition Π is individually stable if no IDS in Π can benefit from shifting from its current community to another one without making the members of the latter community worse off.

Theorem 7. *Any final partition* Π_f *resulting from Algorithm 6 is individual-stable.*

Démonstration. It has already been proven that any Nash-stable case implies individual-stability [95]. Thus, we can conclude that Algorithm 6 converges to individual-stability.

□

Theorem 8. *Algorithm 6 confirms that for any IDS $i \in \mathcal{N}$ that leaves its current community $C_{\Pi}(i)$ and joins another community C_k, the community trust criterion of $C_{\Pi}(i)$ must be greater than the community trust criterion of F_k.*

Démonstration. This can be proven by looking at the condition that makes an IDS i move from its current community C_{Π}(i) and the community C_k. The condition is $U_i(C_k) \succ_i U_i(C_{\Pi_c}(i))$, which ensures that IDS i has a preference over the communities C_k and the current community C_{Π}(i) based on the community trust criterion.

□

7.3.3 Feedback Aggregation

In the previous section, we presented a trust-based community formation model that enables a set of cloud-based IDSs to cooperatively set up their communities. The output of the community formation algorithm (Algorithm 6) is a set of communities, where each community consists of a set of IDSs that prefer to work with each other. In this section, we show how an IDS inside a community can aggregate feedbacks received from other IDSs in the same community. For this purpose, we use the Dempster-Shafer Theory (DST) for feedback aggregation. DST was selected for the following two reasons : (1) unlike other aggregation models (e.g. Bayesian aggregation model) that demand complete information of prior probabilities, DST can handle a lack of complete information (i.e. uncertainty), and (2) it has the property of preventing collusion attacks, which occur when several malicious IDSs collaborate to give misleading judgments [91]. It is worth noting that a trust-based hedonic game which uses DST was first proposed in [91]. Our work uses a similar methodology for aggregating trust. However, the main differences between our work and the aforementioned work is that we never use feedback aggregation to create a community, instead, feedback aggregation is used after a community has been created in order to aggregate feedback received from IDSs. It is also worth noting that DST is regarded as a useful approach in uncertain reasoning and is widely used in trust-based multi-agent applications (e.g., [142] [91] [143] [144] [141]).

In our model, the frame of discernment, which describes the status of a suspicious intrusion (hypothesis) is $\Omega = \{1, 0, U\}$. In this set, 1 means that IDS j decides and reports to IDS i that there is an intrusion, 0 means that IDS j decides and reports to IDS i that there is no intrusion, and U shows that IDS j is uncertain whether there is an intrusion or not. Each

hypothesis is assigned a basic probability value (bpv) between 0 and 1, which is equal to the credibility score believed by the IDS giving the judgement. For example, assume that IDS c believes and reports to IDS a that a suspicious intrusion I is an actual attack, then the bpv for c would be : $m_c(1) = T_{a,c}$, $m_c(0) = 0$ and $m_c(U) = 1\text{-}T_{a,c}$, where $T_{a,c}$ is the trust value of IDS $c \in \mathcal{N}$, which is obtained from previous experiences of IDS a with IDS c, as illustrated in Section 7.3. On the other hand, if IDS c claims that the suspicious intrusion I is not an actual attack, then the bpv for IDS c would be : $m_c(1) = 0$, $m_c(0) = T_{a,c}$, and $m_c(U) = 1\text{-}T_{a,c}$.

DST combines multiple IDSs beliefs under the condition that evidences from different IDSs are independent. For example, if IDS_i wants to combine the belief of two IDSs IDS_1 and IDS_2 over the same frame of discernment Ω, the combined belief of IDS_1 and IDS_2 is calculated as follows [138] :

$$\begin{aligned} &m_{IDS_1}(1) \oplus m_{IDS_2}(1) = \frac{1}{K}[m_{IDS_1}(1)m_{IDS_2}(1)+ \\ &m_{IDS_1}(1)m_{IDS_2}(U) + m_{IDS_1}(U)m_{IDS_2}(1)] \end{aligned} \tag{7.10}$$

$$\begin{aligned} &m_{IDS_1}(0) \oplus m_{IDS_2}(0) = \\ &\frac{1}{K}[m_{IDS_1}(0)m_{IDS_2}(0) \\ &+ m_{IDS_1}(0)m_{IDS_2}(U) + m_{IDS_1}(U)m_{IDS_2}(0)] \end{aligned} \tag{7.11}$$

$$m_{IDS_1}(U) \oplus m_{IDS_2}(U) = \frac{1}{K}[m_{IDS_1}(U)m_{IDS_2}(U)] \tag{7.12}$$

where,

$$\begin{aligned} K = &m_{IDS_1}(1) + m_{IDS_2}(1) + m_{IDS_1}(1) + m_{IDS_2}(U) \\ &+ m_{IDS_1}(U) + m_{IDS_2}(U) + m_{IDS_1}(U) + m_{IDS_2}(1) \\ &+ m_{IDS_1}(U) + m_{IDS_2}(0) + m_{IDS_1}(0) + m_{IDS_2}(0) \\ &+ m_{IDS_1}(0) + m_{IDS_2}(U) \end{aligned} \tag{7.13}$$

Here is an example. Assume the following :
$m_{IDS_1}(1) = 0.75$ $m_{IDS_1}(0) = 0$ $m_{IDS_1}(U) = 0.25$
$m_{IDS_2}(1) = 0.6$ $m_{IDS_2}(0) = 0$ $m_{IDS_2}(U) = 0.4$

by combining the above two belief functions, we can obtain the result as follows :

$belief(1) = (0.75 * 0.6) + (0.75 * 0.4) + (0.6 * 0.25) = 0.9$
$belief(0) = (0 * 0) + (0 * 0.4) + (0 * 0.25) = 0$
$belief(U) = (0.25 * 0.4) = 0.1$
Since $belief(1) > belief(0) > belief(U)$, IDS i will decide that an attack exists.

7.3.3.1 Illustrative Example of Trust Aggregation

This section presents an example that shows how the proposed DST approach allows IDS_1 to decide whether a given IDS is trustworthy or not. This example is based on all of the information shown on Tables 7.1 and 7.2, which indicate the IDSs' judgments on suspicious intrusion I and the credibility scores (bpv) of each IDS believed by IDS_1, respectively.

Table 7.1 IDSs' judgments on suspicious intrusion I.

IDS	**IDS's Judgement on *I***
IDS_3	Not-attack
IDS_4	Not-attack
IDS_5	Attack

Table 7.2 Credibility scores of IDSs believed by IDS1.

IDS	**Credibility (bpv)**
IDS_3	0.34
IDS_4	0.23
IDS_5	0.95

IDS_1 aggregates all IDSs judgements (or beliefs) on the suspicious intrusion I as follows :

First, let's combine the beliefs of the IDSs IDS_3 and IDS_4. We find the *bpv* for IDS_3 and IDS_4 as follows :

$m_{IDS_3}(1) = 0$, $m_{IDS_3}(0)$=0.34, $m_{IDS_3}(U)$ = 1-0.34 = 0.66

$m_{IDS_4}(0) = 0$, $m_{IDS_4}(0)$=0.23, m_{IDS_4}(U) = 1-0.34 = 0.77

Aggregate IDS_3 and IDS_4 judgements :

— $m_{IDS_3}(1) \oplus m_{IDS_4}(1) = \frac{1}{k}[m_{IDS_3}(1)m_{IDS_4}(1) + m_{IDS_3}(1)m_{IDS_4}(U) + m_{IDS_3}(U)m_{IDS_4}(1)]$
Where,
K $= m_{IDS_3}(1)m_{IDS_4}(1) + m_{IDS_3}(1)m_{IDS_4}(U) +$

$m_{IDS_3}(U)m_{IDS_4}(1) + m_{IDS_3}(0)m_{IDS_4}(0) + m_{IDS_3}(0)m_{IDS_4}(U) +$
$m_{IDS_3}(U)m_{IDS_4}(0)+m_{IDS_3}(U)m_{IDS_4}(U)$

K =0 * 0 + 0 * 0.77 + 0.66 * 0 + 0.34 * 0.23 +
0.34*0.77+ 0.77*0.23+0.66*0.77 = 1.0253
$m_{c3}(1) \oplus m_{c4}(1) = \frac{0}{1.0253}$=0

— $m_{IDS_3}(0) \oplus m_{IDS_4}(0) = \frac{1}{K}[m_{IDS3}(0)m_{IDS_4}(0) +$
$m_{IDS_3}(0)m_{IDS_4}(U) + m_{IDS_3}(U)m_{IDS_4}(0)]$
K= 1.0253
$m_{IDS_3}(0) \oplus m_{IDS_4}(0) = \frac{0.4918}{0.8482} = 0.5171$

— $m_{IDS_3}(U) \oplus m_{IDS_4}(U) = \frac{1}{K}[m_{IDS_3}(U) \oplus m_{IDS_4}(U)]$
K = 1.0253

$m_{IDS_3}(U) \oplus m_{IDS_4}(U) = \frac{(0.66*0.77)}{0.8482}$= 0.5991

Then, we combine the aggregated beliefs of IDS_3 and IDS_4's with the beliefs of IDS_5 as follows :

$m_{IDS_{3,4}}(1) = 0$, $m_{IDS_{3,4}}(1) = 0.5171$, $m_{IDS_{3,4}}(U)$= 0.5991

$m_{IDS_5}(1) = 0.95$, $m_{IDS_5}(1)$=0, $m_{IDS_5}(U)$ =0.05

— K = $m_{IDS_{3,4}}(1)m_{IDS_5}(1) + m_{IDS_{3,4}}(1)m_{IDS_5}(U) +$
$m_{IDS_{3,4}}(U)m_{IDS_5}(1) + m_{IDS_{3,4}}(1)m_{IDS_5}(0) +$
$m_{IDS_{3,4}}(0)m_{IDS_5}(U)+$
$m_{IDS_{3,4}}(U)m_{IDS_5}(0) + m_{IDS_{3,4}}(U)m_{IDS_5}(U)$

= 0∗0.95+0∗0.05+0.5991∗0.95+0.5171∗0+
0.5171∗0.05+0.5991∗0.05+0.5991∗0.05 = 0.658045

— $belief(1)$= $m_{IDS_{3,4}}(1) \oplus m_{IDS_5}(1) = \frac{0.569}{0.658}$=**0.864**
— $belief(0)$= $m_{IDS_{3,4}}(0) \oplus m_{IDS_5}(0) = \frac{0.055}{0.658}$= 0.084
— $belief(U)$= $m_{IDS_{3,4}}(U) \oplus m_{IDS_5}(U) = \frac{0.029}{0.658}$= 0.045

Although both IDS_3 and IDS_4 judge that the suspicious intrusion I is "not attack", IDS_1's belief that I is "attack" is still high after combining IDS_3 and IDS_4's belief with IDS_5. The reason is that the credibility of IDS_5 is higher than IDS_3 and IDS_4. This is considered a strong advantage of using the Dempester-Shafer Theory (DST) for trust aggregation.

7.4 Fairness Assurance

In the previous section, we proposed a framework for forming a trust-based cooperative cloud-based IDS. In this section, we show how to achieve the fairness among IDSs inside the community. The purpose of having a fairness assurance mechanism is to prevent the selfish behavior inside the communities. Such *selfish* IDSs frequently send consultation requests to other IDSs and at the same time do not answer other IDSs' consultation requests with the aim of saving their own resources. Thus, for the benefit of the community members, there should be a suitable mechanism that guarantees fairness among the community members. In fact, fairness assurance can also bring the following two advantages. First, it encourages IDSs to participate in the community. Secondly, fairness assurance mechanisms can minimise unnecessary consultation requests, which can be exploited to launch DoS attacks. It is worth mentioning that the methodology used here is inspired by that used in resource allocation [156] ; however, different parameters, constraints and formulas are used in order to make it adequate for trust-based multi-cloud environment and also adequate to model *selfish* and *well-behaving* IDSs.

We formulate in this section the fairness assurance problem as a Stackelberg game [17] between the *well-behaving* IDSs and the *selfish* ones. We denote a *well-behaving* IDS by i and a *selfish* IDS by j. i plays the *leader* of the game and starts the game by selecting its consultation rate c_{ij}, where c_{ij} represents i's consultation request rate, and it depends on the trust value of j with respect to i. The *selfish* IDS j is the *follower* that observes the leader's strategy and chooses its best response to it in terms of response rate denoted by r_{ji}. Table 7.3 summarizes the different notations that are used in this section.

The game is modeled as an optimization problem and the backward induction reasoning is used to find the optimal actions of both the *well-behaving* and *selfish* IDSs. This is achieved by extracting first the best response of j to the observed action of i and then merging this best response to i's optimization problem to help it chose the optimal response rate. This means that the *well-behaving* IDS i expects that the *selfish* IDS will play his best responses to i's consultation rate strategy and adds this information into its optimization problem to chose the optimal response rate strategy.

Table 7.3 Notations

Symbol	Significance
i	Normal IDS
j	Selfish IDS
c_{ij}	Consultation request rate of IDS i to j
r_{ij}	Response rate of IDS i to j' consultation request rate
r_{ji}	Response rate of IDS j to i' consultation request rate
M_i	Total out-bound communication rate of i
M_j	Total out-bound communication rate of j
$\mathcal{N}_j$	A set of IDSs in the same community as j
$\mathcal{N}_i$	A set of IDSs in the same community as i
T_{ij}	the trust value of the *selfish* IDS j with respect to i
$Sat_i(r_{ji})$	the satisfaction of the *well-behaving* IDS i to j's response rate r_{ji}

Let us first fix the *well-behaving* IDS i's policy to a certain policy c_{ij}. c_{ij} represents the consultation rate of i. After observing c_{ij}, j needs to solve the following optimization problem in order to determine its optimal response to c_{ij} :

$$\begin{aligned} & argmax \sum_{i \in \mathcal{N}_j} T_{ij}\, Sat_i(r_{ji}) \\ & Subject\ to: \sum_{i \in \mathcal{N}_j} r_{ji} \leq M_j \\ & r_{ji} \leq c_{ij} \\ & r_{ji} \geq 0 \end{aligned} \tag{7.14}$$

The above optimization problem aims to find the optimal response rate r_{ji} (output) of the *selfish* IDS j to i' consultation request rate. The optimization problem uses the satisfaction of the $well-behaving$ IDS i to j's response rate (Sat_i) and the trust value of the $selfish$ IDS j with respect to i (T_{ij}) as input. $\mathcal{N}_j$ is the set of *well-behaving* IDSs that belong to the same community as j and M_j denotes the maximum communication rate of the *selfish* IDS j. The satisfaction function $Sat_i(r_{ji})$ is defined as follows :

$$Sat_i(r_{ji}) = log_2(1 + \frac{r_{ji}}{c_{ij}}) \tag{7.15}$$

The logarithmic function indicates that the *well-behaving* IDS i becomes more satisfied when j increases its response rate to i's consultation requests. Since the utility function given in (Eq. (7.15)) is strictly convex in $\vec{r_j}$ and the feasible set is convex, the optimization problem can be considered as a convex optimization problem and hence accepts a unique solution.

We can see that when the first constraint ($\sum_{i\in\mathcal{N}_j} r_{ji} \leq M_j$) of the optimization problem (Eq. (7.14)) is an inactive constraint, which happens when M_j is very large, the solution will be equal to $r_{ji} = c_{ij}$. However, when $\sum_{i\in\mathcal{N}_j} r_{ji} \leq M_j$ is an active constraint, the solution can be achieved by forming a Lagrangian function as follows :

$$\begin{aligned}\mathcal{L}^j(\vec{r}_j, \lambda_j, \phi_{ji}, \mu_{ji}) = & \sum_{i\in\mathcal{N}_j} T_{ij}(1 + log\frac{r_{ji}}{c_{ij}}) - \\ \lambda_j(\sum_{i\in\mathcal{N}_|}(r_{ji} - M_j) & - \sum_{i\in\mathcal{N}_|}(\phi_{ji}(r_{ji} - c_{ij})) \\ + \sum_{i\in\mathcal{N}_|} \mu_{ji} r_{ji}) & \end{aligned} \tag{7.16}$$

where λ_j, ϕ_{ji}, $\mu_{ji} \in \mathbb{R}+$ satisfy the complementarity conditions $\lambda_j(\sum(r_{ji} - M_j)) = 0$, $\phi_{ji}(r_{ji} - c_{ij}) = 0$ and $\mu_{ji} r_{ji}$ =0, for all $i \in \mathcal{N}_j$.

The optimization problem in (14) with its constraints is equivalent to finding solutions for the following set of equations :

$$\begin{aligned} & arg \underset{\vec{r}_j}{maximize}\ \mathcal{L}^j(\vec{r}_j, \lambda_j, \phi_{ji}, \mu_{ji}) \\ & Subject\ to : \lambda_j(\sum_{i\in\mathcal{N}_j}(r_{ji} - M_j)) = 0 \\ & \qquad \phi_{ji}(r_{ji} - c_{ij}) = 0,\ \ \forall i \in \mathcal{N}_j \\ & \qquad \mu_{ji} r_{ji} = 0,\ \ \forall i \in \mathcal{N}_j \end{aligned} \tag{7.17}$$

We maximize the above Lagrangian optimization problem and get the first-order Kuhn-Tucker condition :

$$\frac{T_{ij}}{r_{ji} + c_{ij}} = \lambda_j + \phi_{ji} - \mu_{ji} \tag{7.18}$$

When the first constraint ($\sum_{i\in\mathcal{N}_j} r_{ji} \leq M_j$) of problem (14) is active and the second constraint ($r_{ji} \leq c_{ij}$) of the same problem is not, the closed-form solution is supported with the equality condition :

$$\sum_{i\in\mathcal{N}_j}(r_{ji} - M_j) = 0 \tag{7.19}$$

and hence, we get the optimal solation

$$r^*_{ji} := \frac{T_{ij}}{\sum_{v \in \mathcal{N}_j}(T_{vj})}(M_j + \sum_{u \in \mathcal{N}_j} c_{uj}) - c_{ij} \tag{7.20}$$

Let's move now to the *well-behaving* IDS's side (IDS i). IDS i knows that j will play its best response r^*_{ji} to the i's strategy (consultation request rate) c_{ij} and incorporates this knowledge into its optimization problem to determine the solution r_{ij} that maximizes its own payoff. Thus, the IDS i has to solve the following problem :

$$\begin{aligned} &\underset{\vec{r_i}}{maximise} \sum_{j \in \mathcal{N}_i} r_{ij} \\ &Subject\ to: \qquad r_{ij} \leq max(r^*_{ji}, c_{ij}) \\ &\qquad\qquad\quad \sum_{j \in \mathcal{N}_i} r_{ij} \leq M_i \\ &\qquad\qquad\quad r_{ij} \geq 0 \end{aligned} \tag{7.21}$$

The above optimization problem guarantees that i's response rate r_{ij} to j's consultation requests never exceeds the j's response rate r^*_{ji} unless i has more consultation requests to be sent to j. This can clearly be shown using the constraint $r_{ij} \leq max(r^*_{ji}, c_{ij})$. We conclude that i can control its response rate to j's consultation requests.

Proposition 2. *The proposed fairness assurance mechanism can achieve the fairness among IDSs*

Démonstration. In order to behave selfishly, an IDS should frequently send consultation requests and at the same time do not answer other IDSs' consultation requests. In other words, maximizing consultation requests rate and minimizing response rate. The selfish IDS's maximization strategy can be undermined by controlling recipient IDS's response rate using the constraint $r_{ij} \leq max(r^*_{ji}, c_{ij})$ in the optimization problem (21). On the other hand, the selfish IDS's minimization strategy can also be undermined in the optimization problem (14) using the constraint $r_{ji} \leq c_{ij}$. Thus, the selfish IDS's consultation request rates and response rates are controlled by other IDSs (recipient IDSs).

□

7.5 Experimental Evaluation

In this section, we first explain the setup used to do our experimentation and then study the performance of the proposed trust-based cooperative intrusion detection approach.

7.5.1 Experimental Setup

Our model is implemented in a 64-bit Windows 8 environment on a host equipped with an Intel Core i7-4790 CPU 3.60 GHz Processor and 16 GB RAM. We used Matlab for implementing our approach.

In our simulations, we use 100 IDSs. Each IDS is described by two parameters, trust value t and decision threshold τ. The trust value represents the level of the expertise of the IDS, which denotes the ability of the IDS to catch suspicious traces from a given observation. The threshold τ represents the sensitivity of the IDS. Lower values of τ indicate a more sensitive IDS.

We use a Beta density function to reflect the intrusion detection capability of each IDS (same used in [11]). A Beta density function is given by :

$$\begin{aligned} f(z|\alpha,\beta) &= \frac{1}{B(\alpha,\beta)z^{\alpha-1}(1-z)^{\beta-1}} \\ B(\alpha,\beta) &= \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}dx \end{aligned} \tag{7.22}$$

$$\begin{aligned} \alpha &= 1 + \frac{t(1-d)}{d(1-t)}r \\ \beta &= 1 + \frac{t(1-d)}{d(1-t)}(1-r) \end{aligned} \tag{7.23}$$

where $z \in [0, 1]$ is the assessment result from the IDS about the likelihood of intrusion, and $f\ (z|\alpha,\beta)$ is the distribution of assessment z from an IDS with trust level t to an intrusion with difficulty level $d \in [0, 1]$. The trust level in the distribution can represent the expertise level of the IDS. Higher values of d represent these attacks that are difficult to detect. Higher values of t indicate a higher probability of generating correct intrusion assessments. $r \in \{0, 1\}$ is the expected result of detection. $r = 1$ means that there is an intrusion and $r = 0$ means otherwise.

In order to evaluate the ability of the proposed model in the presence of an untrusted environment, we made the percentage of untrusted IDSs 70% (trust level $t \leq 0.2$). We argue, based on the recent literature [139], that the percentage of untrusted nodes tends to form the majority compared to that of trusted nodes. We applied the proposed community formation algorithm (Algorithm 6) on the considered IDSs. We compared the proposed aggregation approach with other known aggregation approaches in the state-of-the-art : Majority aggregation model [62] and the weighted average aggregation model [96]. In the majority model, the IDS collects feedback from IDSs about suspicious behaviour and the decision is made (i.e., attack or not)

according to the majority. However, in the weighted average aggregation model, weights W are assigned to feedbacks from different IDSs to distinguish their detection capability. Highly trusted IDSs are assigned with larger weights compared to low trusted IDSs. The decision is made according to the following equation. If $(\Sigma_{k=1}^{n} W_k y_k) \ / \ (\Sigma_{k=1}^{n} W_k) \geq \tau$, the decision is *the existence of an attack*. Otherwise, the decision is that *there is no attack*, where W_k is the weight of the k-th IDS and y_k is the feedback from the k-th IDS.

7.5.2 Experimental Results

In Figure 7.3, we observe that the proposed Dempster-Shafer aggregation approach shows a significant enhancement for the false negative rate, compared to the majority and weighted feedback aggregation models. The results are given for different threshold values τ. Also, in Figure 7.4, our model yields a significant enhancement for the false positive rate, compared to the majority and weighted feedback aggregation models. This is due to the fact that the proposed feedback aggregation (i.e., Dempster-Shafer) ignores the untrustworthy feedbacks while building the final decisions. Moreover, Dempster-Shafer puts a weight for each feedback according to the level of the trustworthiness of the IDS giving this feedback.

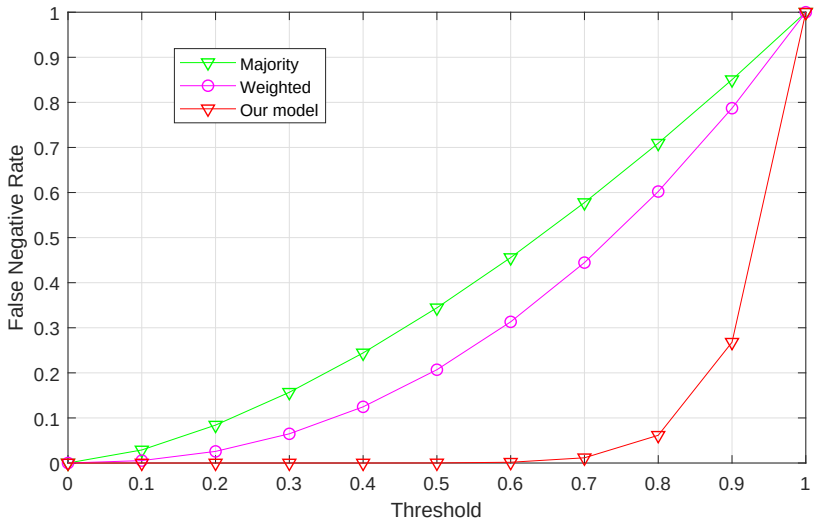


Figure 7.3 False Negative Rate : Comparison of three aggregation models.

In Figure 7.5 and Figure 7.6, we also study the impact of the expertise level (i.e, trust value) on the detection accuracy. To achieve that, we execute the proposed trust-based community formation algorithm (Algorithm 6) many times. Each time, we allow IDSs to have different values of t. This experiment is performed with different threshold values τ. Figure 7.5 shows

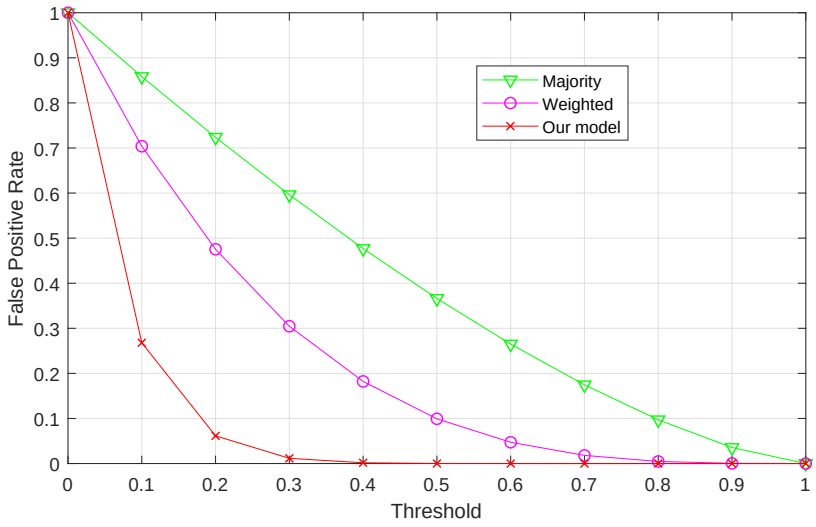


Figure 7.4 False Positive Rate : Comparison of three aggregation models.

that the false negative rate decreases when the level of the trustworthiness of an IDS increases. Also, Figure 7.6 shows that the false negative rate decreases when the level of the trustworthiness of an IDS increases. This is due to the fact that whenever an IDS becomes more trusted, the probability to give the right feedback about suspicious intrusions increases.

Figure 7.7 provides comparison between the proposed coalitional game model and the trust-based Grand community approach. The latter considers all existing IDSs during the cooperation. In other words, the community contains all IDSs. Therefore, the feedback is received from all IDSs and the final decisions are made using the Dempster–Shafer aggregation approach. This is unlike our model where we first run a trust-based community formation Algorithm (Algorithm 6) and reduce the number of IDSs inside the community. The result shows that the proposed model yields an improvement for both the false positive and false negative rates compared to the trust-based Grand community approach. This is due to the fact that the proposed approach reduces the rate of untrusted IDSs inside the community. Therefore, the received feedback is more likely to reflect the real status of any suspicious intrusion, whether it is a real attack or not. However, for the Grand community approach, the feedback is received from all IDSs. Thus, the chance of receiving incorrect feedback increases. We also study the cost associated with using each approach in Figure 7.7. The cost here means the time needed to make a judgment about a suspicious intrusion. The result is projected in a range between 0 and 1. The proposed approach gives a low overhead compared to the Grand community approach. The is justified by the fact that our approach reduces unnecessary consultation requests. Only those trusted IDSs are consulted. Unlike in the Grand

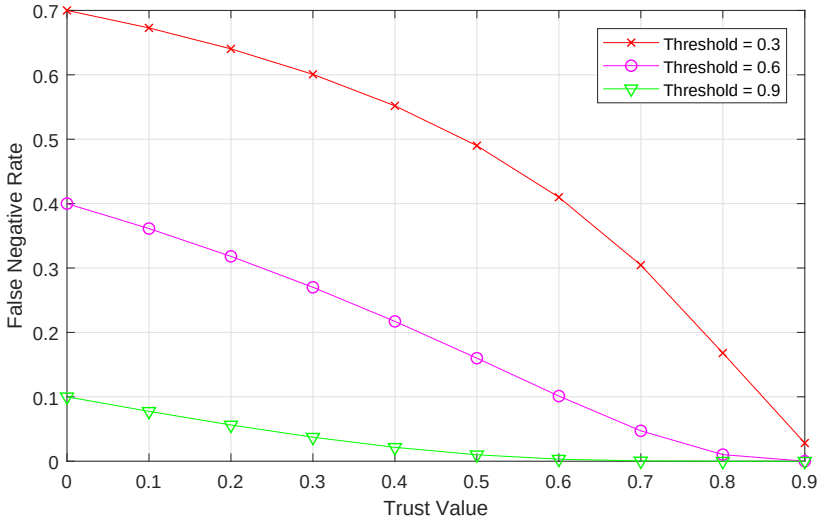


Figure 7.5 False Negative with the variations of Trust Values.

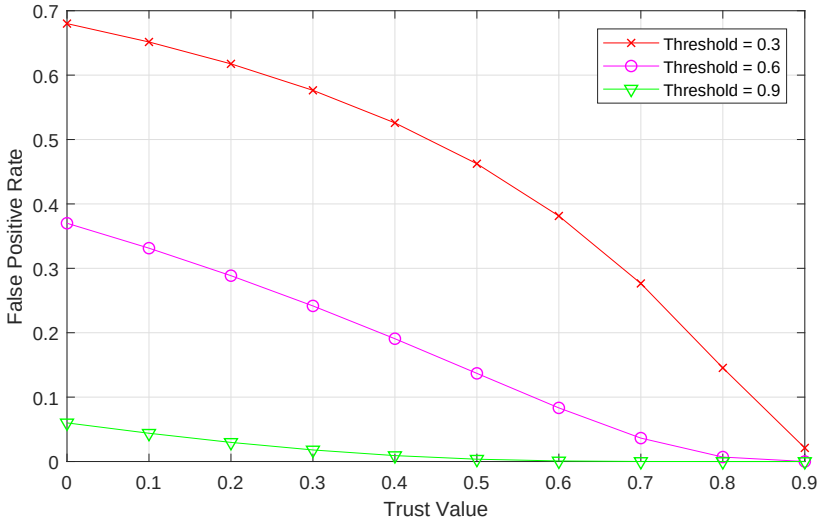


Figure 7.6 False Positive with the variations of Trust Values.

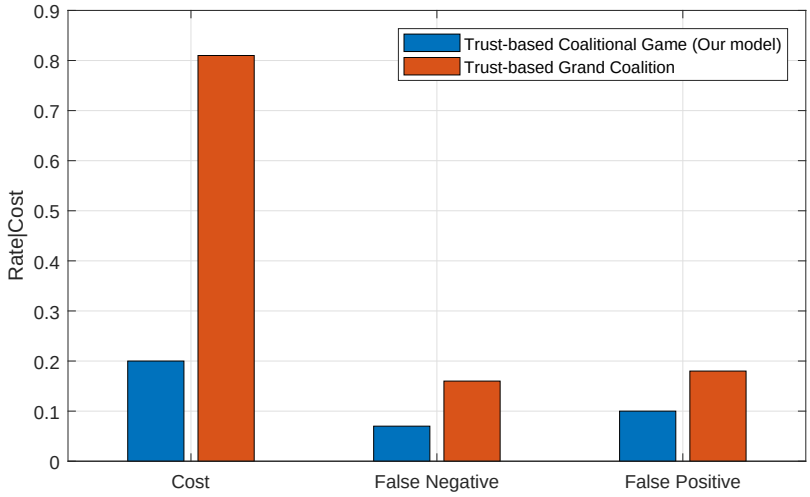


Figure 7.7 Comparison of two community formation models.

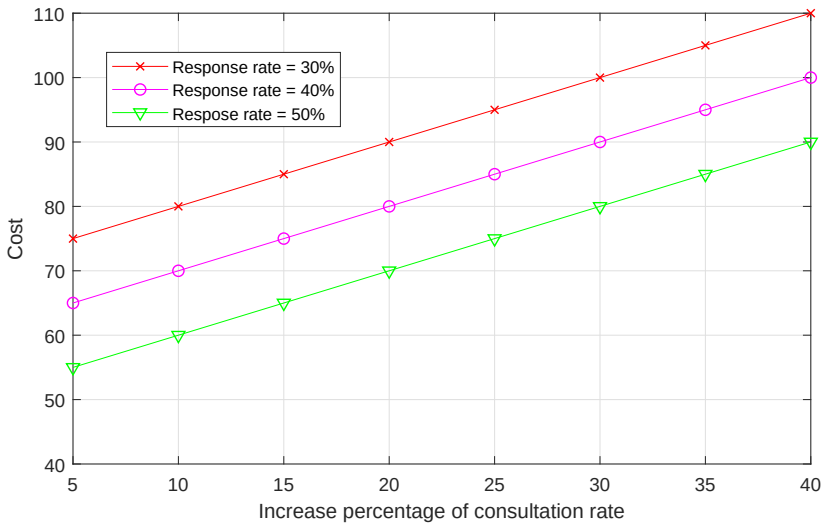


Figure 7.8 Cost with regards to the increase in the percentage of consultation rate.

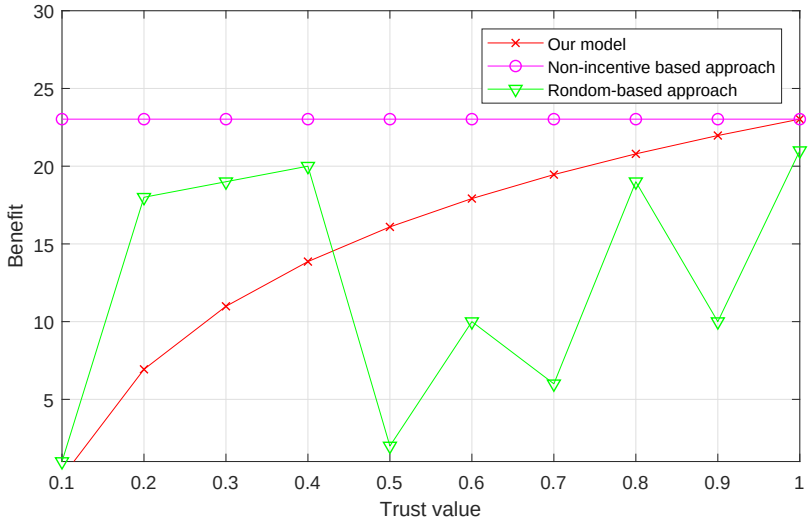


Figure 7.9 Benefit with regards to the trust value.

community approach where a consultation about suspicious intrusion is sent to all available IDSs.

To evaluate the proposed fairness assurance mechanism, we made some IDSs to behave selfishly by assuming that they will not answer other IDSs' consultation requests, while at the same time they keep sending consultation requests. In Figure 7.8, we show how the proposed fairness assurance mechanism can be used to minimize the selfish behavior. Figure 7.8 shows the cost (i.e., time) incurred by increasing the *selfish* IDSs' consultation rates. The cost here represents the average amount of consultation requests that have never been answered by the receiving IDSs. The results show that the selfish IDSs have no incentive to increase the percentage of consultation requests in such a way that exceeds the amount specified by *well-behaving* IDSs (according to equations 14 and 21). The study is presented to different normal IDSs' response rates. We can see from the figure that this cost increases as the selfish IDSs' consultation rates increase from 5% to 40%. The linear relationships indicates that any given change in the selfish IDSs' consultation rate will always produce a corresponding change in the cost.

In Figure 7.9, we compare the proposed Stackelberg fairness assurance model with two other approaches, namely random and non-incentive ones. The random-based approach enables IDSs to set the response rates given to other IDSs randomly, regardless of the latter's trust values. However, in a non-incentive approach, the response rates are equally assigned to each IDS regardless of their trust values as well. Our approach performs differently by assigning response rates to other IDSs based on their trust values. Obviously, Figure 7.9 shows that our

model enables IDSs having higher trust values to gain a better chance to get more benefits. The benefit here describes the percentage of consultation requests answered by the consulted IDSs. This indicates that the proposed model is incentive-compatible, since it gives privileges to those IDSs that have high trust values to obtain more benefits, which encourages them to participate in the community formation process. The results show also that the random-based approach is neither fair nor incentive-compatible since it allows IDSs with low trust values to receive more benefits than those IDSs having high trust values. For example, the IDS with a trust value 0.2 received benefits of 18, which is higher than those having trust values of 0.6 (that have yielded benefits amounting to 10). Also, the non-incentive based approach is not incentive-compatible since the benefit distribution to IDSs is uniform without giving privileges based on IDSs' trust values.

7.6 Conclusion

This paper investigates a novel trust-based cooperative IDS in a multi-cloud environment. To this end, we propose a cooperative game-theoretical framework. The framework enables an IDS to evaluate the trust values of other IDSs using bayesian inference. We devise also a community formation algorithm, based on the coalitional game theory. The algorithm enables IDSs to join and/or leave a given community in such a way that enhances their ability to work with trusted IDSs. The proposed algorithm converges to a Nash-stable situation ; that is, no IDS has an incentive to leave its current community and move to another one. Furthermore, we propose a trust aggregation algorithm, based on the Dempster-Shafer Theory (DST), to enable an IDS inside a community to aggregate feedbacks about suspicious attacks in order to make the optimal decision in terms of detection accuracy. In addition, we formulate a fairness assurance mechanism as a Stackelberg game between the *well-behaving* IDSs and the *selfish* ones (that frequently send consultation requests to other IDSs and at the same time do not answer other IDSs' consultation requests). Experimental results show that the proposed fairness assurance mechanism can encourage IDSs to participate in the community and reduce unnecessary consultation requests that can be exploited to launch DoS attacks. Moreover, the results show the effectiveness of the proposed approach in terms of minimizing false positive and false negative rates, and minimizing the time needed to judge suspicious attacks. Finally, the results reveal the effectiveness of the proposed fairness mechanism in terms of achieving fairness among IDSs in terms of benefits obtained through cooperation.

CHAPTER 8 ARTICLE 5 : A DEEP LEARNING APPROACH FOR PROACTIVE MULTI-CLOUD COOPERATIVE INTRUSION DETECTION SYSTEM

Adel Abusitta, Martine Bellaïche, Michel Dagenais, and Talal Halabi
Future Generation Computer Systems (Submitted).

Abstract

The last few years have witnessed the ability of cooperative cloud-based Intrusion Detection Systems (IDS) in detecting sophisticated and unknown attacks associated with the complex architecture of the Cloud. In a cooperative setting, an IDS can consult other IDSs about suspicious intrusions and make a decision using an aggregation algorithm. However, undesired delays arise from applying aggregation algorithms and also from waiting to receive feedback from consulted IDSs. These limitations render the decisions generated by existing cooperative IDS approaches ineffective in real-time, hence making them unsustainable. To face these challenges, we propose a machine learning-based cooperative IDS that efficiently exploits the historical feedback data to provide the ability of proactive decision making. Specifically, the proposed model is based on a Denoising Autoencoder (DA), which is used as a building block to construct a deep neural network. The power of DA lies in its ability to learn how to reconstruct IDSs' feedback from partial feedback. This allows us to proactively make decisions about suspicious intrusions even in the absence of complete feedback from the IDSs. The proposed model was implemented in GPU-enabled TensorFlow and evaluated using a real-life dataset. Experimental results show that our model can achieve detection accuracy up to 95%.

8.1 Introduction

The complex architecture of cloud computing systems make them vulnerable to many kinds of attacks. Recently, promising results have shown that the use of cooperative Intrusion Detection Systems (IDSs) can improve the detection accuracy compared with the traditional single IDSs [67] [11] [12]. This is due to the fact that it is becoming largely difficult for a single IDS to detect all existing attacks [11] [12], due to its limited knowledge of such attack patterns and implications. The cooperation among IDSs that belong to different Cloud Providers (CPs) can be achieved by allowing them to exchange their intrusion analysis feedback and exploit

each other's expertise to cover unknown threat patterns, thus achieving mutual benefits.

Unfortunately, there are considerable delays associated with the use of existing cooperative IDSs. These delays are mostly due to the computation complexity of using the aggregation algorithms such as Dempster-Shafer Theory (DST), and also the large geographic distances that might separate the IDSs. In fact, each IDS, after receiving feedback from consulted IDSs about a suspicious intrusion, is required to use a suitable feedback algorithm, in order to make a final decision about the suspicious intrusion. The aggregation technique is usually costly in terms of computation time and depends on many factors such as the number of consulted IDSs, the IDSs' expertise and trust levels [11] [12]. Moreover, due to uneven IDSs' connections and Internet speeds and other unknown factors (e.g., busy IDSs, compromised IDSs), there is no guarantee that feedback will be received simultaneously. Thus, decisions on whether or not to rise an alarm about suspicious intrusions might be excessively delayed due to the missing feedback of a single IDS. Hence, the decisions generated by the cooperative IDS are ineffective in a real-time setting, making it unsustainable.

To overcome these limitations, we design a proactive multi-cloud cooperative IDS that integrates a machine learning-based approach. The proposed model exploits the historical feedback to predict the status of suspicious intrusions. This can be done proactively without having to apply any aggregation method on consulted IDSs' feedback, nor having to wait until receiving all the feedbacks from the consulted IDSs, i.e., only partial or incomplete feedback can be used to predict the status of suspicions intrusions. This, in turn, makes our solution reliable and feasible in real-time environments, where decisions about intrusions must be taken rapidly in order to effectively apply the required action measures at the right time. However, due to the structure of used data, which are collected from many IDSs with different expertise and trust levels, and due to the fact that decisions regarding suspicious intrusions must be taken despite the existence of missing feedback, the use of traditional machine learning techniques (e.g., SVM) for such a problem produces inaccurate results in terms of prediction accuracy [157][158]. For this purpose, we propose a deep learning approach that consists of multiple layers to learn the representation of the data with multiple levels of abstraction [157][158]. This allows us to learn how to obtain a "good" representation of the data to be used later as inputs to supervised machine learning techniques in order to achieve better detection accuracy [159] [160] [161] [18][19].

More particularly, our model is based on Stacked Denoising Autoencoders (SDAE), where a denoising autoencoder is used as a building block to train a deep network [18][19]. Our model exploits the fact that a denoising autoencoder can learn how to reconstruct original inputs giving partial data inputs, by allowing deep neural networks to learn (during unsupervised

pre-training stage) how to extract features that are robust to incomplete IDSs' feedback. Such robust features can be seen as useful representations of data to yield a better intrusion detection accuracy in such real-time environments. This makes our detection model proactive on two levels : (1) by making decisions about suspicious intrusions even with missing feedback, and also (2) by making decisions without having to apply any aggregation method on consulted IDSs' feedback. Our contributions are summarized as follows :

— Proposing a cooperative intrusion detection model (based on stacked denoising autoencoders) that enables making decisions about suspicious intrusions even with partial IDS's feedback. This, in turn, accelerates the decision making in real-time environments.
— Designing a proactive multi-cloud cooperative IDS, which allows us to make decisions about suspicious intrusions proactively, i.e., without the need to apply aggregation methods on IDSs' feedback.
— Proposing an approach to extract robust features that yield a better performance in cooperative intrusion detection.
— Evaluating the effectiveness of the proposed solution using a real-life dataset, and comparing our results with those generated with other approaches.

Our model has been implemented in GPU-enabled TensorFlow and evaluated using a real-life dataset. The results show the effectiveness of the proposed approach in terms of enhancing the accuracy of the detection compared to the state-of-the-art deep architectures : Multi-Layer Perceptron (MLP) and Stacked Auto Encoders (SAE).

The remainder of this paper is organized as follows. In Section 8.2, the related work is introduced. The proposed cloud-based cooperative intrusion detection system is presented in Section 8.3. Experimental results show the effectiveness of the proposed approach in Section 8.4. Finally, Section 8.5 concludes the paper.

8.2 Background and Related Work

Cloud-based IDSs can be divided into two categories : signature-based and anomaly-based [1]. The former compares suspicious activity with known attack patterns. To make signature-based IDSs effective, the database of signatures should be updated repeatedly. On the other hand, anomaly-based IDSs raise alarms when unexpected behaviours have been detected. Anomaly-based IDSs are efficient in detecting unknown attacks. A database of known attacks is not required for this approach. However, the shortcoming of this ssolution lies in the relative high false positive rate compared with the signature-based technique [11]. In fact, IDSs may

use both techniques to enhance their detection accuracy. However, the detection accuracy is limited by the amount of information held by the IDSs. Recent research has shown that the collaborative detection can enhance the detection rate up to 60% [11] [23]. This section presents the state-of-the-art of the cloud-based cooperative IDSs.

In multi-cloud cooperative IDSs, Lo et al. [62] propose a detection approach that exchanges alerts among different nodes (i.e., hosts) whenever an attack is detected. They adopt a rule-based technique to detect SYN attacks by fetching the threshold for rule patterns through the initial rule establishment phase. The advantage of this approach is that it is can be used to distribute the detection overhead between the nodes. Also, Teng et al. [61] proposed an approach that aggregates two detectors : a feature detector and a statistical detector. The former uses SNORT to separate events based on transmission control protocols such as TCP. The later cooperates with the former by using packets from it to decide whether an event is an attack or not. A predefined threshold is set for this purpose. An attack is considered in cases where the rate of packets obtained exceeds the threshold.

Deep learning approaches for intrusion detection system were recently proposed in several works [162] [163] [164] [165] [166][167] [168]. However, these approaches have the same problem of single IDS, as they cannot detect all existing attacks [11] [12] due to their limited knowledge about all attack patterns and implications.

Man and Huh [63] and Singh et al. [64] designed a cloud-based cooperative IDS between different regions. The model allows alerts to be exchanged from multiple detectors. Also, knowledge are allowed to be exchanged between interconnected clouds. Ghribi [65] proposed a middle-ware IDS that allows a cooperation between different layers : Hybervisor-based, Network-based and VM-based IDS. If an intrusion is found in a layer, the propagation of such intrusion to the other layers could be prevented. Moreover, Chiba et al. [66] designed a network-based cooperative IDS, which is used to identify network intrusions in the cloud environment. This can be achieved through monitoring network traffic while maintaining and/or guaranteeing Quality of Services (QoS) and performance [66].

In a multi-cloud environment, Dermott et al. [67] proposed a cooperative intrusion detection in multi-cloud environments. The Dempster-Shafer theory of evidence is used to collect and aggregate the beliefs received by the monitoring entities. The received beliefs are used to make the final decision regarding a possible intrusion. This approach has a limitation : its centralized-based architecture, whereby a trusted third-party is responsible for aggregating feedback and managing cloud-based intrusion detection system.

A trust-based cooperative IDS was proposed for a non-cloud environment. For example, Fung and Zhu [11] designed trust-based cooperative IDS. Through cooperation, a local Intrusion

Detection System (IDS) can detect new attacks that may be unknown to other IDSs. They use the diagnosis from different IDSs to perform intrusion detection. They present a system architecture of a collaborative IDS in which trustworthy feedback aggregation is a key component [11]. Also, Zhu et al. [70] [71] designed an incentive-based communication protocol, which provides IDS nodes incentives to send feedbacks to their peers thus preventing malicious behaviors. The limitation of the existing trust-based cooperative IDS is that they consider a consultation request to be sent to a large number of IDSs in order to get a feedback. However, this approach causes extra overhead as some IDSs are needlessly consulted. We address this point by allowing an IDS to make a decision regarding suspicious intrusions without having to wait until receiving all the feedbacks from the consulted IDSs. Also, Abusitta et al.[12] propose a framework that allows cloud-based IDSs to distributively form trustworthy IDSs communities. For this purpose, they propose an algorithm based on the cooperative game theory : it allows a set of IDSs to cooperatively set up their coalition in such a way to allow their individual detection accuracy increase, despite the presence of untrusted IDSs.

Overall, for a multi-cloud environment, a proactive cooperative IDS has yet to be designed. This proactive approach is useful in real-time environments, where fast decisions must be taken. In this work, the proactive aspect was achieved in two ways : 1) making decisions about suspicious intrusions proactively without having to apply aggregation methods on IDSs' feedback ; and 2) making decisions about suspicious intrusions in the absence of complete IDS feedback.

8.3 The Proposed Proactive Multi-cloud Cooperative IDS

8.3.1 System Model

In multi-cloud cooperative IDS, a cloud-based IDS can consult other cloud-based IDSs about suspicious intrusions and aggregate the received feedback to make a decision using an aggregation algorithm. Fig. 8.1 shows the high level architecture of the proposed cooperative IDS. As illustrated, CP4 has a list of several CPs which are open to cooperation. CP4 sends a consultation request to its whitelist for intrusion diagnosis. The recipient IDSs send their feedback (attack or not) to the CP4's IDS, which then uses an aggregation method such as the Dempster-Shafer Theory (DST) [16] for feedback aggregation.

In this paper, we are looking for a proactive approach in order to reduce delays associated with the architecture of multi-cloud cooperative IDS. These delays mostly come from the large geographic distances that might separate cloud-based IDSs and the computation complexity of using the aggregation algorithms, especially when the number of consulted IDSs is large

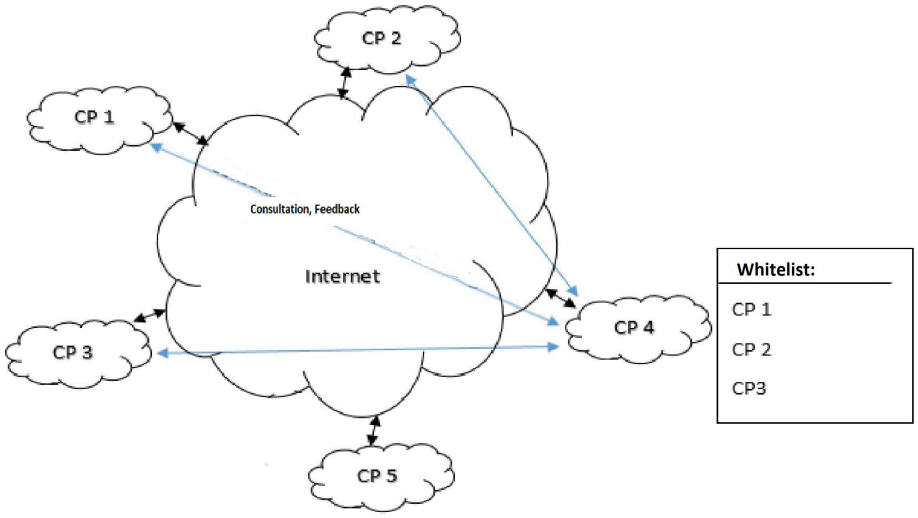


Figure 8.1 Architecture of the proposed cooperative IDS

[11] [12]. To this end, we propose a proactive multi-cloud cooperative IDS that is based on machine learning approach. The proposed model exploits the historical feedback to predict the status of suspicious intrusions. In this solution, decisions can be taken proactively without having to apply an aggregation method on the received feedback an it eliminates waiting time to obtain all the feedbacks from the IDSs.

The subsequent subsections will present the proposed model. We first present the concept of traditional autoencoders followed by the proposed feedback-based denoising autoencoders, where we will show how to train an autoencoder using unsupervised data to enable the extraction of the features that are robust to incomplete feedback. Then, we will introduce the stacked denoising autoencoders and fine-tuning process and show how to stack denoising autoencoders in order to build deep neural networks to be used to proactively make decisions about suspicious intrusions.

8.3.2 The Traditional Autoencoders

An autoencoder is an unsupervised learning approach that is used to learn efficient and reliable data codings [169]. It is used to pre-train each layer in a deep neural network in order to obtain better initial weights that lead to a better performing classification [161]. Researchers have seen that weights initialization using autoencoders can improves the performance of deep neural networks compared to random initialization [161].

The structure of an autoencoder is shown in Fig. 8.2. The dimensions for both input (IDSs'

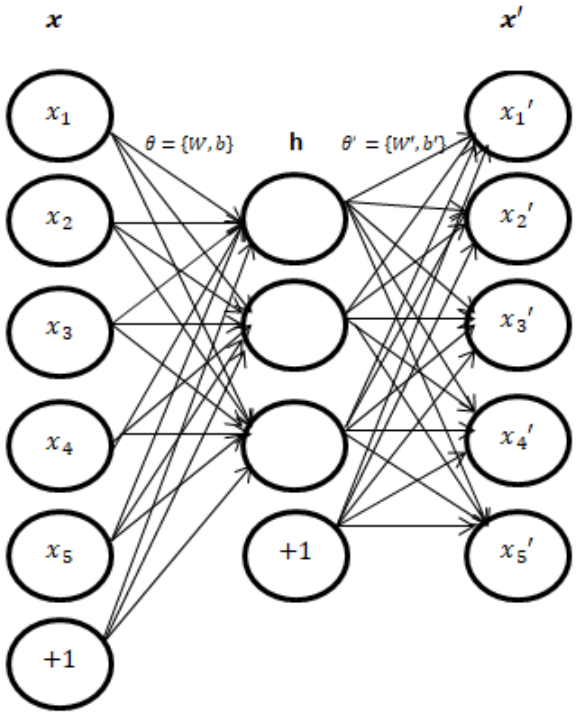


Figure 8.2 Example of an autoencoder.

feedback) and output (reconstruction of IDSs' feedback) are the same as shown in the figure. An autoencoder is used as a building block for deep networks [161]. In particular, it takes input vector (IDSs' feedback) $x \in [0,1]^d$, where d is the vector dimension, and maps it to a hidden representation $h \in [0,1]^{d'}$ using the following equation :

$$h = f_\theta(x) = Sigmoid(W * x + b) \tag{8.1}$$

$\theta = \{W, b\}$, W is a weight matrix and b is a bias vector. After that, the resulting hidden layer representation h will be reconstructed to the output layer x' using a decoding function as follows :

$$x' = g_{\theta'}(h) = Sigmoid(W' * h + b') \tag{8.2}$$

$\theta' = \{W', b'\}$, W' and b' are a weight matrix and a bias vector of the reverse mapping, respectively. The weight matrix W' of the reverse mapping may optionally be constrained by $W' = W^T$, in which case the autoencoder is said to have tied weights [161] [19]. Each training x is thus mapped to a corresponding h and a reconstruction x' .

The purpose of the model is to optimize the parameters (θ and θ') of the model, so that the reconstruction error between input and output can be minimized. The following optimization problem is used for this purpose :

$$\begin{aligned} \theta^*, \theta'^* &= arg\ \ min_{\theta,\theta'} \frac{1}{n}\sum_{i=1}^{n} L(x^{(i)}, x'^{(i)}) \\ &= \frac{1}{n}\sum_{i=1}^{n} L(x^{(i)}, g_{\theta'}(f_\theta(x^{(i)}))) \end{aligned} \tag{8.3}$$

where L is a loss function. Since the input vector x is a binary vector (an IDS's feedback is either 0 or 1), a reconstruction cross-entropy is used as a loss function [170]. Thus, the loss function will be described as follows :

$$L(x, x') = -\sum_{i=1}^{d}[x_i log x'_i + (1 - x_i) log(1 - x'_i)] \tag{8.4}$$

8.3.3 The proposed IDS-based Denoizing Autoencoders

In order to make an autoencoder robust to the incomplete input (IDSs' feedback) and also to prevent it from just learning the identity of the input, the autoencoder should be trained to reconstruct its IDS's feedback even if the feedback does not represent the whole IDSs' feedback (some feedback are not available). The autoencoder that deals with corrupted version of input is called a denoising autoencoder [19]. This can be achieved by adding noise to the initial input x before passing it to the hidden layer in order to reconstruct x, where x are IDSs' feedback. Thus, a partially corrupted version z will be obtained from x as follows :

$$z = qD(x) \tag{8.5}$$

Where qD is a corruption process [19]. In our model, we use Masking Noise (MN) for the corruption process as it is considered a useful method to represent incomplete IDS's feedback [18]. In MN noise, a fraction v (selected at random) of each input x is forced to 0, while the others remain untouched. In fact, other noise can also be used (e.g., Gaussian noise). However, MN is more useful to simulate incomplete IDSs' feedback [18] since the noise will change only partial feedback.

The traditional autoencoder is then used to take corrupted data z and try to learn how to reconstract x. This is done by allowing the input z to be mapped to a hidden representation as :

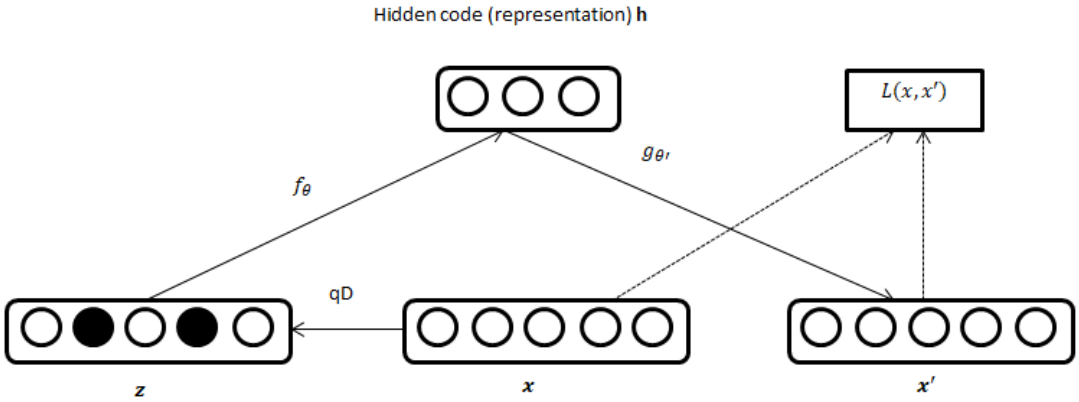


Figure 8.3 IDS-based denoising autoencoder architecture.

$$h = f_\theta(z) = Sigmoid(W' * z + b') \tag{8.6}$$

Note that we selected z as input instead of x as a traditional autoencoder was used. The result of the above equation h is then used to reconstruct x' as follows :

$$x' = g_{\theta'}(h) = Sigmoid(W * h + b) \tag{8.7}$$

The denoising autoencoder architecture is described in Fig. 8.3. As given in the traditional autoencoder, the parameters are trained to minimize the average reconstruction error :

$$\begin{aligned} \theta^*, \theta'^* &= arg \ \ min_{\theta,\theta'} \frac{1}{n} \sum_{i=1}^{n} L(z^{(i)}, x'^{(i)}) \\ &= \frac{1}{n} \sum_{i=1}^{n} L(z^{(i)}, g_{\theta'}(f_\theta(z^{(i)}))) \end{aligned} \tag{8.8}$$

The above equation is then re-written by considering cross-entropy as loss function :

$$L(z, x') = -\sum_{i=1}^{d} [z_i log x'_i + (1 - z_i) log(1 - x'_i)] \tag{8.9}$$

The training algorithm of the proposed IDS-based denoising autoencoder is presented in Algorithm 7. As we can see in the algorithm, for the raw inputs x, randomly selected parts of them will be set to 0 as corrupted inputs z. The corrupted input z will be then encoded to the hidden code and then reconstructed to the output. However, at this point, x' is a

deterministic function of z rather than x. The reconstruction is computed between z and x is denoted by $L(x, x')$ (Same as with the autoencoder). The parameters of the model are initialized randomly and then optimized by stochastic gradient descent algorithms.

Algorithm 7: IDS-based Training Denoising Autoencoder

procedure TRAINING-DA(x, l, e, b, θ)
— x= $[x_1, x_2, ...x_n]$: Input IDSs' feedback
— l : learning rate
— e : Amount of epoches to be iterated
— b : Amount of batches
— θ=$\{W, b, b_h\}$
for *i from 0 to e* **do**
  for *j from 0 to b* **do**
    — $z = CorruptInput(x, c)$: c is corruption level
    — $h = s(z * W + b)$
    — $x' = s(h * W^T + b_h)$
    — $L(x, x')$= $-\sum_{i=1}^{d}[x_i log x'_i + (1 - x_i) log(1 - x'_i)]$
    — $cost = mean(L(x, x'))$
    — g=compute the gradients of the cost w.r.t θ
  for θ_i,g_i $\in$ *(θ,g)* **do**
    — $\theta_i = \theta_i$ - $l * g_i$
  end
  end
  end
end procedure

8.3.4 The proposed IDS-based Stacked Denoising Autoencoders

An autoencoder is used as a building block for unsupervised training of deep networks [161]. Such an architecture is called as Stacked AutoEncoder (SAE) [161]. The purpose of having an autoencoder as a building block to the deep network is to support the pre-training process [161], which is used to initialize the parameters of the deep network before applying supervised learning algorithms. Although Restricted Boltzmann Machine (RBM) (RBM)[171] can also be used as a building block for deep networks, SAE is considered much simpler and easier than RBM [161].

The initialization of deep networks parameters using SAE leads to a better classification accuracy compared to Multi-Layer Perceptron (MLP) [172], which does not use a pre-training process (all of the parameters are initialized randomly). In SAE, each layer's input is obtained from the previous layer's output. Fig. 8.4 shows the first step of a SAE. It trains an autoencoder on raw input x to learn h_1 by minimizing the reconstruction error $L(x, x')$. Once the parameters W_1 and W'_1 of the autoencoder are obtained using the gradient descent algorithm [173], a new layer can be added, as shown in Fig. 8.5. In Fig. 8.5, the hidden representation

h_1 will be an input to train the second autoencoder by minimizing the reconstruction error $L(h_1, h'_1)$. This figure clearly shows that the error is calculated between the output h'_1 and the previous latent feature representation h_1.

Once the parameters W_2 and W'_2 of the second autoencoder are obtained using the gradient descent algorithm, the new hidden representation h_2 will become the raw input for the next autoencoder. These steps can be repeated until the last hidden layer (the output of the last autoencoder) is reached. To determine the number of layers required to build a IDS-based deep neural network, we keep adding layers until no improvement has been achieved in the test error [18] [19] [161]. The last hidden layer represents a "good" representation of data that can be used as input for any supervised learning algorithm [161] [174].

The procedure to train a deep network using the proposed IDS-based denoising autoencoders as a building block (Fig. 8.6) is similar [18] [19]. Its only difference resides in the fact that each layer is trained, i.e., to minimize the criterion in Eq. (8.8) instead of Eq. (8.3). Note that the corruption process qD is used only during training, but not to propagate representations from the raw input to higher-level representations. Note also that when the layer k is trained, it receives as input the uncorrupted output from the previous layers. In Fig. 8.6, after training the first block of denoising autoencoder as shown in Fig. 8.3, the learned encoding function f_θ is used on clean input x as shown in Fig. 8.6 (left). The resulting representation is used to train the second block of denoising autoencoder as shown in Fig. 8.6 (right), to learn the second layer encoding function $f_\theta^{(2)}$. The process can be repeated for the next layers (Fig. 8.7).

The pre-training process is shown in Algorithm 8 : once the mapping function f_θ is learnt using Algorithm 7, the function will be used on IDSs' (complete) feedback to generate the values of the second layer (first hidden layer). These values will then be used as inputs to train the next layer (using Algorithm 7) to generate the second mapping function $f_\theta^{(2)}$. This function will be used to generate the values of the third layer (second hidden layer). These values will also be used as inputs to train the next layer (using Algorithm 7) to generate the third mapping function $f_\theta^{(3)}$. The same steps can be applied for the whole set of hidden layers.

8.3.5 The proposed IDS-based Fine-tuning and Detection

Once the last hidden layer is trained as shown in the previous section, the parameters (generated from Algorithm 8) are used to initialize deep networks. The deep network is now ready to apply supervised machine learning such as SVM or a logistic regression. This can be done by adding a predictor to the last layer. In this work, we use a logistic regression

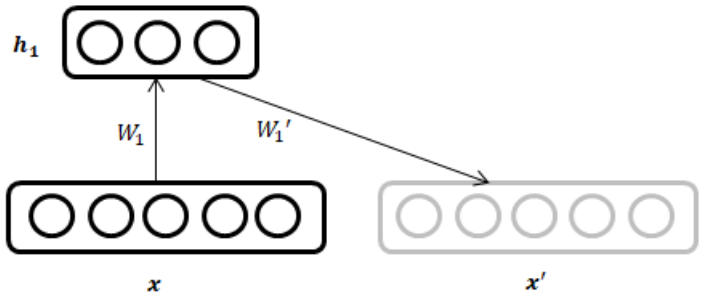


Figure 8.4 Step 1 in Stacked Denoising Autoencoders.

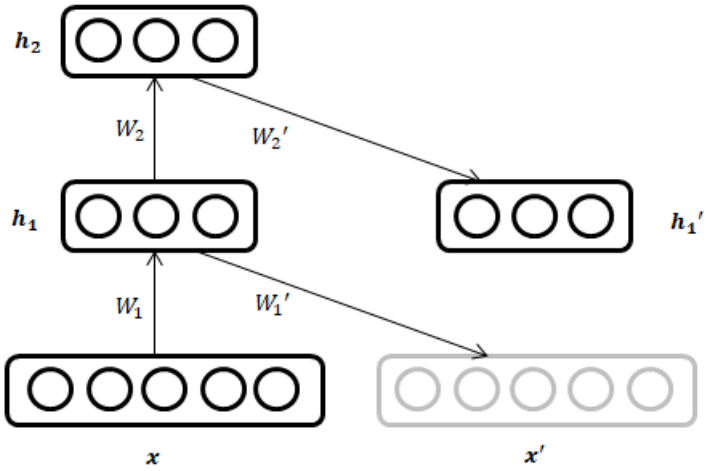


Figure 8.5 Step 2 in Stacked Denoising Autoencoders.

solution as it can lead to better results in binary classifications [175]). To this end, a layer of logistic regression should be added, as shown in Fig. 8, to generate a *deep neural network*. The parameters of the all layers will then be fine-tuned to minimize the error to predict the supervised status (i.e., attack or not) using back-propagation algorithm. The fine-tuning algorithm is depicted in Algorithm 9.

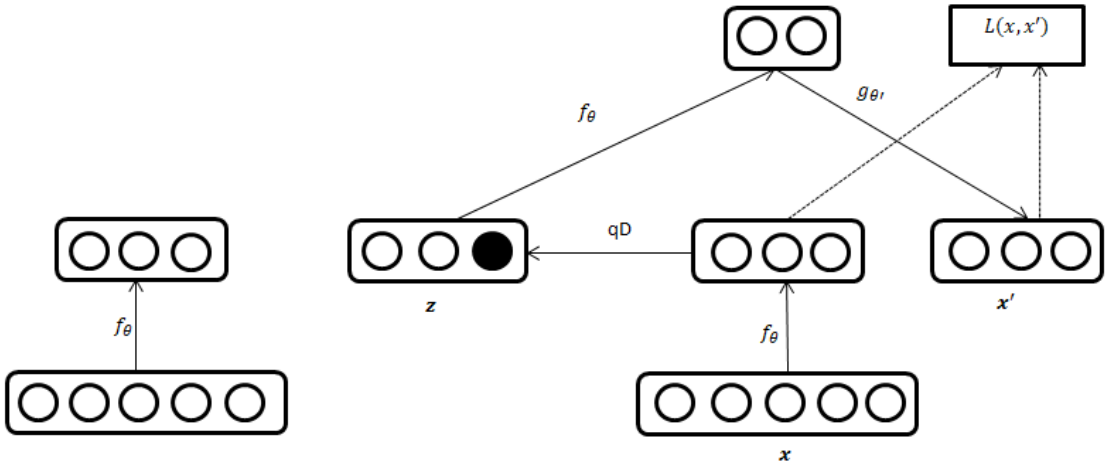


Figure 8.6 Stacking denoising autoencoders. on the left, the encoding function f_θ, which has been learnt in Fig. 8.2, is used on clean input x. on the right, The resulting representation is used to train a second block denoising autoencoder.

Algorithm 8: IDS-based Unsupervised Pre-Training Algorithm

procedure PRE-TRAINING(x,l,e,b,h,Θ)
— x= $[x_1, x_2, ...x_n]$: Input IDSs' feedback
— h=$[h_1, h_2, ...h_z] \in Z^l$
— $\Theta = [\theta_1, \theta_2, ...\theta_z]$,
— Where $\theta_i = \{W_i, b_i, b_{hi}\}$
— O=$[O_1, O_2, ...O_l]$ is the output of each hidden layer, where O_i=$[o_{i,1}$,$o_{i,2}$, ... $o_{i,n}]$
— θ_1 = Training-DA (x, l, e, b, θ_1)
for *i from 1 to n* **do**
— $o_{1,i}$=$Sigmoid(x_i$*W_1+$b_i)$
for *j from 2 to l* **do**
— θ_j = Training-DA(O_{j-1},l,e,b,θ_j)
for *i from 0 to n* **do**
— $o_{j,i}$=$\sigma(o_{j-1}$*W_j+$b_j)$
end
end
end
end procedure

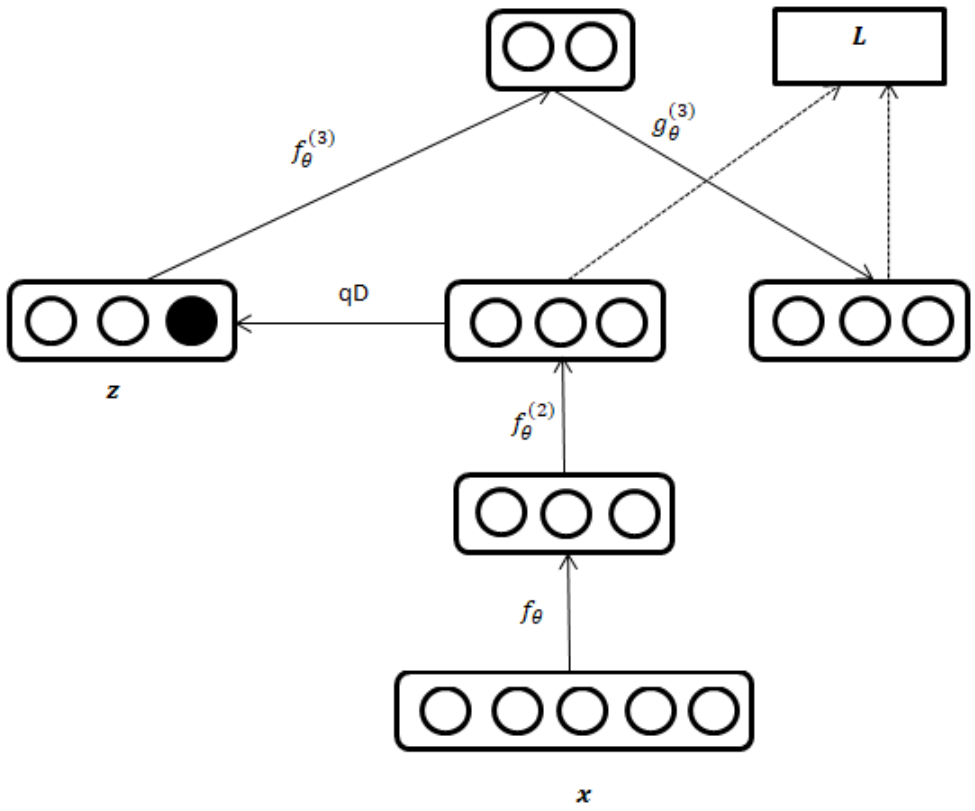


Figure 8.7 The process is repeated for the third block denoising autoencoder.

Algorithm 9: IDS-based Fine Tuning Algorithm

procedure FINETUNING(x,l,e,b,h,Θ)
— x= $[x_1, x_2, ...x_n]$
— h=$[h_1, h_2, ...h_z] \in Z^l$
— $\Theta = [\theta_1, \theta_2, ...\theta_z]$, Where $\theta_i = \{W_i, b_i, b_{hi}\}$
— O=$[O_1, O_2, ...O_l]$: output of each hidden layer, where O_i=$[o_{i,1}$,$o_{i,2}$, ... $o_{i,n}]$
— θ_1 = Training-DA (x, l, e, b, θ_1)
for *epoch from 0 to e* **do**
— cost = $\frac{1}{|D|} = L(\theta = W, b, D)$
— g = compute the gradient of cost w.r.t θ
for θ_i,$g_i \in$ (θ,g) **do**
— $\theta_i = \theta_i$ - $l * g_i$
end
if *validationLoss* < *bestvalidationLoss* **then**
— bestEpoch=epoch
— bestParameters=θ
— bestvalidationLoss=validationLoss
end
end
return *bestParameters*
end procedure

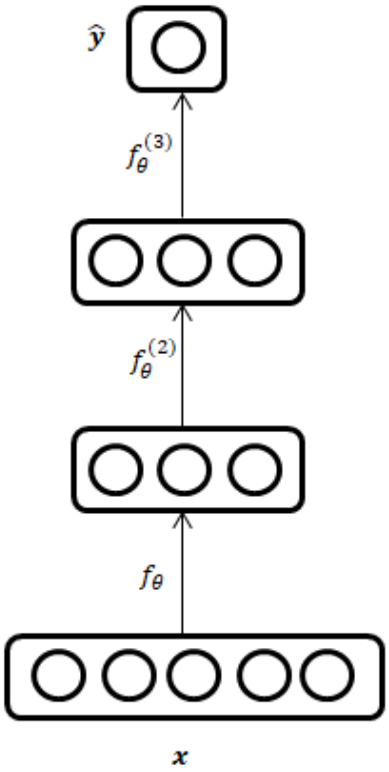


Figure 8.8 The complete architecture of the proposed IDS-based deep neural network after adding the last layer.

8.4 Experimental Evaluation

This section first describes the setup used to perform the evaluation. Then, the performance of the proposed proactive multi-cloud cooperative intrusion detection system is examined.

8.4.1 Experimental Setup

The proposed detection model was implemented using TensorFlow. The evaluation was performed using GPU-enabled TensorFlow running on a 64-bit Windows 8 with an Intel Xeon 3.60 GHz processor, 16 GB RAM. To evaluate this model, a dataset containing IDSs' feedback about suspicious intrusions was required. An IDS feedback about a given suspicious intrusion will be either 1 (if considered as an attack) or 0 (if not). This dataset was created based on the KDD Cup 99 dataset, where each 1 or 0 in the new dataset corresponds to the answer of an ID to a given row of the KDD Cup 99 dataset [176].

Once the dataset was created, it was used to train the model. Then, the ability of the proposed model in making decisions about suspicious intrusions was tested, even in the presence of partial/incomplete feedback. To represent partial/incomplete IDSs' feedback, some of the IDSs' feedbacks (selected randomly) were left blank. In this case, blanks indicate that some of IDSs' feedbacks have yet to be received, due to unexpected delays (busy IDSs, compromised IDSs, etc.).

The model (SDAE-IDS) was trained to create dataset using the 10-fold cross-validation model. Table 8.1 shows the parameters used for the experiment.

Table 8.1 Experimentation parameters.

parameter	considered values
number of cloud-based IDSs N	70
number of hidden layers h	$\{1, 2, 3\}$
number of units per hidden layer	$\{70, 140, 210, 280, 350\}$
corrupting noise level v	30%
learning rate	0.05
output layer	binary logistic regression

8.4.2 Experimental Results

The accuracy of the proposed model was first examined and compared with having all the IDSs' feedback (complete information). This is important to evaluate the effectiveness of the proposed model in making decisions given partial or incomplete feedback. Figure 8.9

shows that the average accuracy of the proposed model, with a variety of hidden units (ranging from 70 to 350), was slightly degraded (less than 1.5%). These results suggest that the proposed machine leaning-based approach can effectively make the right decisions about suspicious intrusions even in the absence of complete feedback from consulted IDSs. Next,

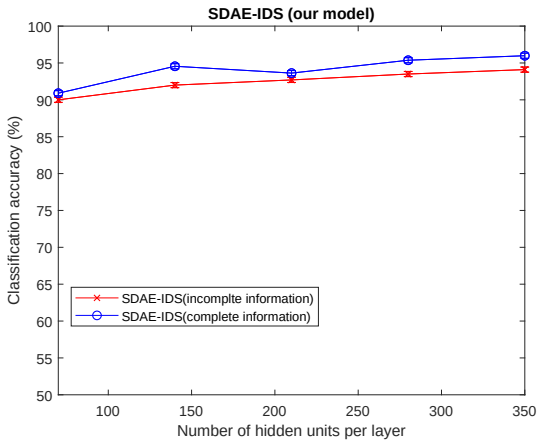
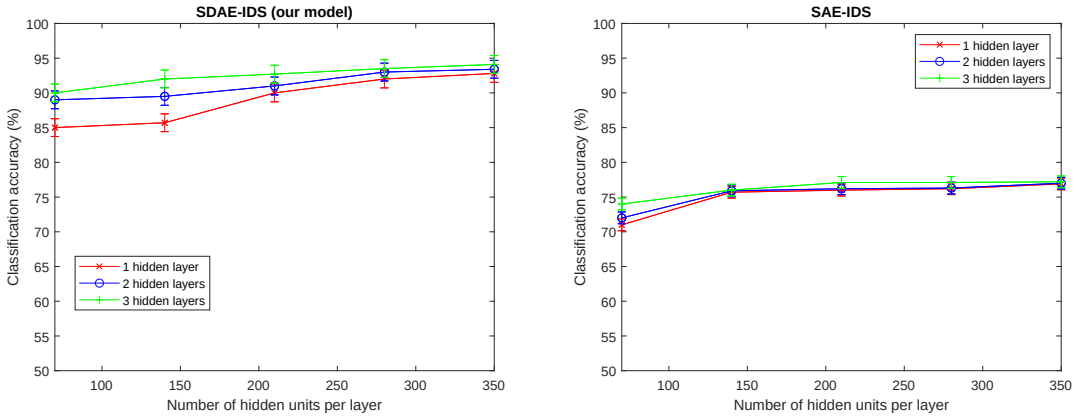


Figure 8.9 Classification accuracy performance compare to having all the IDSs' feedback (complete information) - number of hidden layers = 3.

in Fig. 8.10, this model (i.e., SDAE-IDS) was compared with another approach, namely Stacked Auto Encoder-IDS (SAE-IDS). SAE-IDS uses traditional autoencoders (illustrated in Section 8.3.2) as a building block for the deep neural networks. As illustrated in Section 8.3, in our model (SDAE-IDS), denoising autoencoders are used as building blocks for the deep neural network. This figure illustrates the classification accuracy to make decisions regarding suspicious intrusions in the absence of complete feedback from the IDSs. The study was



(a) (b)

Figure 8.10 Classification accuracy performance for SDAE-IDS (left) and SAE-IDS (right). Error bars show 95% confidence intervals.

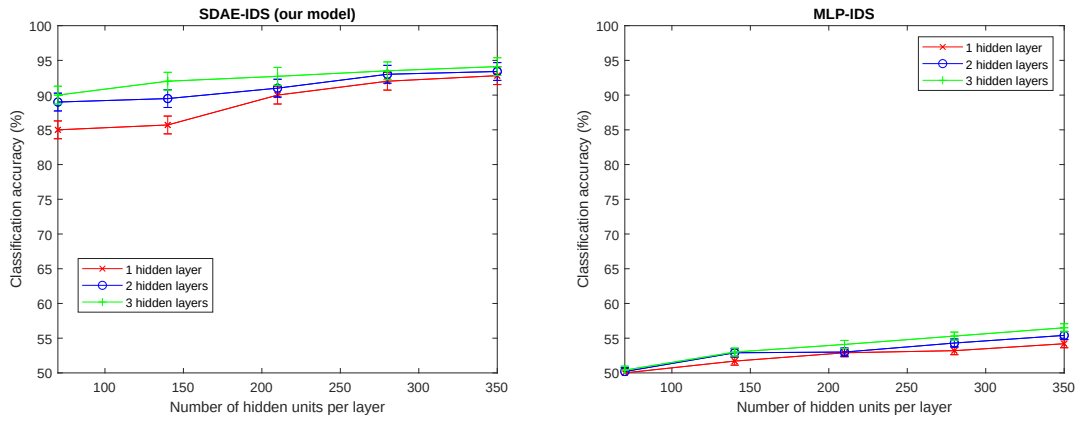
conducted with different numbers of layers and hidden nodes. As shown in Fig. 8.10, our model yields increased accuracy compared to SAE-IDS. The study was done by considering three layers (the same number of layers is considered to study the deep neural network in [18]). More specifically, as for 1-hidden layer, Fig. 8.10a shows that the average accuracy obtained by the proposed model at different numbers of hidden units (ranging from 70 to 350) is 87.5%. This result is better than the results obtained using 1-hidden layer in SAE-IDS (72.50%) as shown in Fig. 8.10b. Also, our model yields a improved accuracy compared to SAE-IDS using 2-hidden layers and 3-hidden layers. More particularly, Fig. 8.10a shows respectively that the 2-hidden layers' average accuracy and 3-hidden layers' average accuracy obtained by the proposed model at different numbers of hidden units (ranging from 70 to 350) are 91.5% and 92.5%. This result is better than the results obtained using SAE-IDS (75.50% for 2-hidden layers and 76.5% for 3-hidden layers) as shown in Fig. 8.10b.

Clearly, for both approaches, SDAE-IDS and SAE-IDS, accuracy and the number of layers increase proportionally : this can be interpreted by the fact that each added layer allows for more representative data to be used during classification. Moreover, as the number of hidden units increases, the classification accuracy is enhanced. This is due to the fact that more hidden units allow more features to be captured from the data, hence enhancing the accuracy of the detection.

The reason why SDAE-IDS (our model) yields a better accuracy than SAE-IDS is that it uses denoising autoencoders as a building block for deep neural networks. Denoising autoencoders allow the deep neural network to extract robust features that lead to a better classification despite the incomplete feedback given as inputs to the deep neural network [19]. The denoising autoencoder learned how to reconstruct the feedback from corrupted input. This enables it to generate a "good" and useful representation despite corrupted input (missing feedback) thus enhancing the classification. This is not the case with SAE-IDS, as a basic autoencoder is used as a building block, which is unable to generate useful data from corrupted input.

Figure 8.11 denotes the detection accuracy, comparing our model (i.e., SDAE-IDS) with multi-layers perceptron (MLP)-IDS. MLP-IDS uses a deep neural network without pre-training process. Furthermore, the study was conducted with different layers and hidden nodes. As for 1-hidden layer, Fig. 8.11a shows that the average accuracy obtained by the proposed model at different numbers of hidden units (ranging from 70 to 350) is 87.5%. This result is better than the results obtained using 1-hidden layer in MLP-IDS (52.20%) as shown in Fig. 8.11b. Also our model yields improved accuracy compared to MLP-IDS using 2-hidden layers and 3-hidden layers. More particularly, Fig. 8.11a shows respectively that the 2- and 3-hidden layers' average accuracy obtained by the proposed model with different numbers

of hidden units (ranging from 70 to 350) are 91.5%. and 92.5%. This result is superior to those obtained using MLP-IDS (53.20% for 2-hidden layers and 53.50% for 3-hidden layers) as shown in Fig. 8.11b. The reason is that our model uses pre-training process which allows the deep network to have better initialization of parameters to be used then during backpropagation and fine tuning. Figures 8.12, 8.13, and 8.14 denote the test classification error with



(a) (b)

Figure 8.11 Classification accuracy performance for SDAE-IDS (left) and MLP-IDS(right). Error bars show 95% confidence intervals.

different numbers of layers. As for 1-hidden layer, Fig. 8.12 shows that the average classification error obtained by the proposed model at different numbers of hidden units (ranging from 70 to 350) is 10.40%. This result is better than the results obtained using 1-hidden layer in SAE-IDS (24.80%) and MLP-IDS (46.50%). Also, our model yields enhanced accuracy compared to SAE-IDS and MLP-IDS using 2-hidden layers and 3-hidden layers. In particular, Figs. 8.13 and 8.14 respectively show that the 2- and 3-hidden layers' average accuracy obtained by the proposed model with different numbers of hidden units (ranging from 70 to 350) are 9.8%. and 7.20%. This result is superior to those obtained using SAE-IDS (24.70% for 2-hidden layers and 23.5% for 3-hidden layers) and MLP-IDS (46.10% for 2-hidden layers and 45.5% for 3-hidden layers). Moreover, Figures 8.12, 8.13, and 8.14 indicate that the test classification error decreases as the number of hidden units increases. This is due to the fact that more hidden units allow for more features to be captured from the data, thus enhancing the accuracy of detection. Figures 8.15 and 8.16 compare SDAE-IDS (our model) with 3-hidden layers and two other denoising approaches [177] based on training with noisy input, namely SAE(1)-IDS and SAE(2)-IDS. SAE(1)-IDS is a 3-hidden-layers SAE-IDS where noisy inputs were only used for the pretraining [18]. SAE(2)-IDS is also 3-hidden-layers SAE-IDS where noisy inputs were used for both pretraining and fine-tuning [18]. These results demonstrate that our framework is also resilient to the increase in the percentage of noises. Figures 8.15

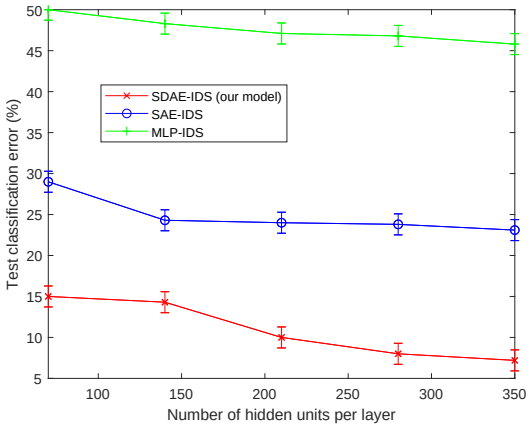


Figure 8.12 Test classification error (%) - 1 hidden layer.

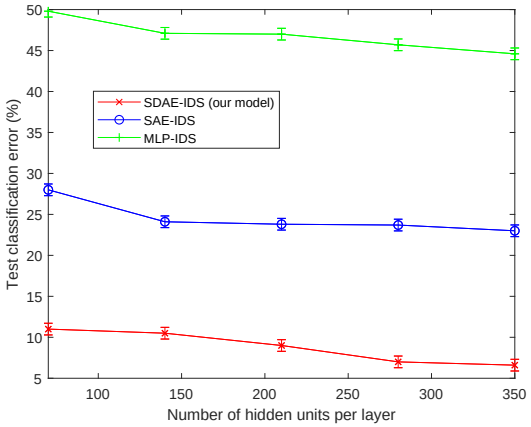


Figure 8.13 Test classification error (%) - 2 hidden layers.

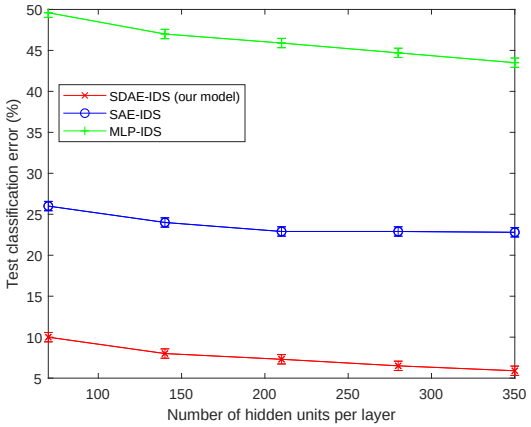


Figure 8.14 Test classification error (%) - 3 hidden layers.

and 8.16 respectively show that the accuracy and test classification error obtained by the proposed model at different percentages of corruption (from 0 % to 40%) are 89.2% and 9.8%. These results are better than those obtained using SAE(1)-IDS (69.7% for accuracy and 30.3% for classification accuracy) and SAE(2)-IDS (65.4% for the accuracy and 34.6% for test classification error). Note that when the percentage of corrupted inputs equals 0%, the three models (SAE(1)-IDS, SAE(2)-IDS and SDAE-IDS) yield the same results in terms of accuracy and error rate due to the fact that when 0% is applied, the three stacked models will be the same as SAE [18].

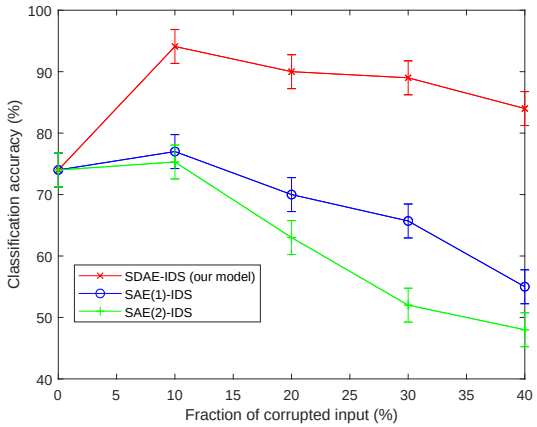


Figure 8.15 SDAE-IDS vs. training with noisy input. Hidden layers have 350 units each

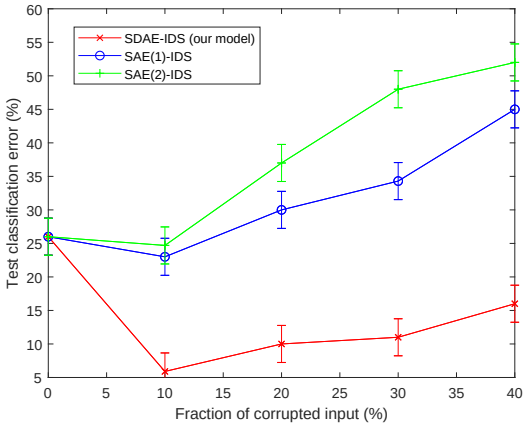


Figure 8.16 SDAE-IDS vs. training with noisy input. Hidden layers have 350 units each

8.5 Conclusion

In this paper, we proposed a proactive multi-cloud cooperative IDS. The proposed model allows us to exploit the historical received feedback to produce learned models used for predicting the status (attack or not) of suspicious intrusions. The proposed model is based on stacked denoising autoencoders, where we use a denoising autoencoder as a building block for deep learning. The proposed IDS-based denoising autoencoder is used to learn how to reconstruct original IDSs' feedback given incomplete IDSs' feedback. This, in turn, allows us to learn how to extract features that are robust to incomplete feedback. Such robust features enables a useful representation of data, enabling a better performing cooperative intrusion detection. The proposed model allows making decisions regarding suspicious intrusions despite the existence of partial or incomplete IDSs' feedback. Also, the proposed model can make decisions regarding suspicious intrusions without having to apply an aggregation method on the consulted IDSs' feedback. Our model was implemented in GPU-enabled Tensor-Flow and evaluated using a real-life dataset. The results show the efficiency of the proposed approach in terms of enhancing the cooperative intrusion detection accuracy compared with the existing state-of-the-art deep architectures.

CHAPTER 9 GENERAL DISCUSSION

In this chapter, we first recall the research objectives that we declared at the beginning of the thesis and then illustrate to which extent these objectives were achieved. Thereafter, we discuss the limitations of our work.

9.1 Objectives achievement

The main objective of this thesis was to enable the detection of cyber attacks in the cloud under complex, dynamic and heterogeneous environments, and to also enable the detection under limited and/or incomplete information about intrusions. More specifically, the following objectives were attained :

— **Enabling intrusion detection in the cloud under changing and heterogeneous environments.** In Chapter 4, we presented a framework to achieve this objective. The proposed framework consists of two important tracks. In the first track, we monitor and analyse the effect of changing and heterogeneous environments on the collected data, in order to remove irrelevant run-time details. To this end, we propose an algorithm based on a low overhead monitoring based tool (i.e., LTTng). The algorithm allows us to determine to which extent the collected data have been affected with respect to the changes applied (e.g., granting/revoking resources to/from the VMs). The output of the algorithm is a filter that reflects to which extent the parameters/metrics are affected due to the changes that have occured. This filter is then used before the prediction process, to get rid of the "noise" that may show up on the collected data (e.g., due to the new changes). The filtering process enables us to remove the effects and irrelevant run-time details from the data in order to provide a robust and generic data to be used to enhance the detection at any environment. The proposed framework paved the road for a generic IDS in the cloud.

— **Creating trustworthy multi-cloud cooperative IDS.** This objective has been achieved in Chapter 5. We proposed a framework for achieving a trust-based cooperative IDS in a multi-cloud environment. We devised a Bayesian-based algorithm for this purpose. In this algorithm, when a cloud-based IDS consults another IDS regarding a suspicious attack, the received feedback and the revealed result (i.e., attack or not) are used to update the trust value of the consulted IDS. We devised a bootstrapping mechanism to determine the initial trust values. The trust value can be promoted if the IDS successfully diagnosed the consultation request about a suspicious intruder

and it can be demoted otherwise. The trust value here represents and shows the accuracy of the IDS diagnosing suspicious attacks. Thereafter, we devised a novel coalition formation algorithm, that is based on the coalitional game theory. The algorithm enables IDSs to set their preferences and allow them to leave or join a given coalition in such a way that enhances their ability to work with trusted IDSs. We proved that the proposed algorithm converges also to a Nash-stable situation ; that is, no cloud-based IDS has an incentive to leave its current coalition to move to another coalition.

— **Enabling trust-based feedback aggregation.** Also, in Chapter 5 we proposed a trust-based feedback aggregation. The output of the coalition formation algorithm (illustrated above) is a set of coalitions, where each coalition consists of a set of cloud-IDSs that prefer to work with each other. The proposed feedback aggregation method enables an IDS inside a coalition to aggregate feedbacks received from other IDSs based on the DST for feedback aggregation. The proposed aggregation method has two advantages : (1 unlike other aggregation models, such as the Bayesian aggregation model, that demand complete information of prior probabilities, DST can handle a lack of complete information (i.e. uncertainty), and (2 it has the property of preventing collusion attacks, which occur when several malicious IDSs collaborate to give misleading judgments.

— **Creating trustworthy federated clouds.** This objective has been achieved in Chapter 6. To this end, we proposed a framework for the formation of trustworthy clouds. Two approaches were proposed to evaluate the CPs trust values : objective and subjective trust evaluations. In the former, we proposed an objective trust evaluation model based on Bayesian inference. In this model, the trust value is evaluated based on the history of interactions (i.e. experience). In the latter, we devised a subjective trust evaluation method based on the DST of evidence integrated with the Bayesian inference. The proposed model allows us to evaluate the trust value in the absence of previous interactions. Thereafter, we devised a novel a federation formation algorithm, based on the coalitional game theory, that allows a set of CPs to cooperatively set up their federations in order to maximise the trust of the formed federations. We proved also that the proposed federation formation algorithm converges also to a Nash-stable situation, i.e. no CP has a motivation to go out from its current federation and move to another federation.

— **Enabling fairness-assurance in multi-cloud cooperative IDS.** In Chapter 7, we proposed a fairness assurance mechanism, in order to prevent (or at least minimize the probability of) well-behaving cloud-based IDS to deal with selfish IDSs when forming a community. The proposed fairness assurance mechanism is modeled as a Stackelberg

game in which each well-behaving cloud-based IDS plays as the leader of the game and the selfish cloud-based IDS is the follower. The strategy of the selfish cloud-based IDSs is to maximise their consultation rates and at the same time minimise their response rates. The well-behaving IDSs know this strategy and choose the optimal response rates that are fairly compatible with their consultation rates. The optimization problem was solved using the backward induction reasoning, through binding at the beginning the best response of the selfish IDS to the well-behaving IDSs consultation rate strategy, and then merging this information into the well-behaving IDSs optimization problem. The outcome of the game is the optimal response rate for the well-behaving IDS. We also proved that the proposed fairness assurance mechanism can achieve fairness among cloud-based IDSs

— **Enabling proactive multi-cloud cooperative IDS.** This objective has been achieved in Chapter 8. We proposed a deep learning detection approach that consists of multiple layers to learn the representation of the data with multiple levels of abstraction. This allowed us to learn how to obtain a "good" representation of the data, to be used later as inputs to supervised machine learning techniques, in order to achieve better detection accuracy. The proposed deep learning model is based on Stacked Denoising Autoencoders (SDAE), where a denoising autoencoder is used as a building block to train a deep network. The proposed model exploited the fact that a denoising autoencoder can learn how to reconstruct the original inputs, given incomplete data inputs, by allowing the deep neural networks to learn (during unsupervised pretraining stage) how to extract features that are robust to incomplete IDSs feedback. Such robust features can be seen as useful representations of data to yield a better intrusion detection accuracy in real-time environments, where decisions about intrusions need to be taken fast, to effectively apply the required measures at the right time.

9.2 Limitation

While it is desirable to use realistic trust values for our implementation, we were unable to find a dataset that contains IDS and CPs trust values. For this purpose, we adopted two approaches to calculate trust values for IDSs and CPs. As for IDSs, we used a Beta density function to reflect the intrusion detection capability of each IDS. As for CPs, we experimentally derived trust values using Cloudsim. To simulate an untrusted non-malicious CPs, we made the CP accept more tasks (i.e. referred as Cloudlets in Cloudsim) in such a way to largely exceed its capacity. Thus, the CP becomes unable to achieve other CPs Cloudlets

within the expected time. On the other hand, in order to simulate a malicious CP, we made the CP intentionally remove resources (i.e. VMs) given to other CPs. Then, we followed the Bayesian inference to derive trust values. For each CP, a default trust value was set. For each Cloudlet it receives, the new trust value is computed from the old one according to the regularized incomplete beta function. The trust value is promoted if the cloud achieves the task successfully (i.e. the task is accepted and achieved within the expected time) and it is demoted otherwise. The trust value for each cloud is calculated after performing several Cloudlets.

Another limitation of the proposed solution is the context of simulated (i.e., Cloudsim) cloud federations. While it may be desirable to implement the proposed model using an open source cloud management system such as OpenStack, CloudStack or OpenNebula, we preferred to use Cloudsim due to the setup needed to validate the model on a large scale, with 100 CPs. Each CP is supposed to provide a computing power similar to a public cloud (e.g., Amazon). This large-scale setting cannot be easily achieved in a local setup. Moreover, it cannot be achieved using public clouds either, because of some restrictions and regulations regarding large-scale testing [2]. This is why many research groups (e.g., [148], [149] [150]) use Cloudsim in their setup. Cloudsim allows the simulation of realistic large-scale CPs and the study of federations and their corresponding policies in terms of jobs migration, automatic scaling and reliability of services/applications [148], [149] [150].

CHAPTER 10 CONCLUSION AND RECOMMENDATIONS

In this thesis, we proposed a new framework for intrusion detection in the cloud that addresses 1) the detection under heterogeneous and changing environments and 2) the detection under limited and incomplete information about attacks. We conducted a literature review to ensure the originality of our methodologies and solutions and to ensure filling the gap in the state-of-the-art work. In particular, we designed a generic cloud-based IDS that allows the detection under changing and heterogeneous cloud environments. Experiments conducted on a real dataset show that the proposed solution enhances the detection, compared to other detection approaches, under largely changing environments. Second, we proposed the first trust-based cooperative IDS in multi-cloud environments. Experiments reveal that the proposed model largely reduces the number of untrusted (malicious or not) cloud-based IDSs in the communities, and dramatically enhances the detection accuracy compared to the existing approaches. Third, we advanced the first fairness-assurance mechanism between the well-behaving cloud-based IDSs and the selfish ones that frequently send consultation requests and do not answer other IDSs consultation requests, with the aim of saving their own resources. Experiments show that the proposed mechanism enables well-behaving IDSs to play the optimal strategy that minimises the chances of cooperating with selfish IDSs, that frequently send consultation requests and not answer other IDSs consultation requests, with the aim of saving their own resources. Finally, we proposed a proactive multi-cloud cooperative IDS that can be efficiently used in real-time environments. Experiments show that the proposed model can effectively make decisions about suspicious intrusions, even in the absence of complete feedback from the IDSs.

The following points summarize the main contributions of this thesis :

— We proposed a framework for achieving a generic cloud-based IDS in order to detect intrusions or attacks under complex, changing and heterogeneous environments.
— We designed a framework that enables a low-overhead monitoring and analysis of changes (e.g., resource scaling) in the cloud in order to understand the effects of these changes on the data used for IDS. We proposed an algorithm that allows us to filter out these effects and remove irrelevant run-time details from the collected data, in order to provide a robust feature that can then be integrated into the proposed detection algorithm.
— We proposed a trust-based multi-cloud cooperative IDS. We introduced the first trust framework that enables a cloud-based IDS to evaluate other cloud-based IDSs trust-

worthiness using Bayesian inference. Thereafter, we devised a novel algorithm, based on a hedonic game theoretical model, that is combined with Bayesian inference. The proposed algorithm allows us to establish trustworthy cloud-based IDS communities and to ensure Nash stability, that is no community member has an incentive to leave its current community and join another one.

— We proposed a new approach for achieving the trust-based cooperation among CPs. Thus, a CP can outsource some of its workloads to other CPs. This, in turn, serves a dual purpose, reducing the extra overhead during the monitoring and detection process, and secondly exploiting the power of other cloud-based IDS in handling sophisticated and severe attacks, since other clouds may have better investments (in terms of hardware and software security) in their intrusion detection solutions.

— We devised a trust-based aggregation method for preventing collusion attacks in the federated cloud-based IDSs. We elaborated a Dempster-Shafer theory-based approach which enables a cloud-based IDS to prevent collusion attacks, which occur when several malicious IDSs collaborate to give misleading judgments.

— We proposed a fairness-assurance multi-cloud cooperative IDS. We designed a fairness-assurance mechanism based on the Stackelberg game, between the well-behaving cloud-based IDSs and the selfish ones that frequently send consultation requests, and do not answer other IDSs consultation requests, with the aim of saving their own resources. The proposed model enables a cloud-based IDS to play the optimal strategy that minimises the chances of joining and cooperating with selfish IDSs.

— We proposed a proactive multi-cloud cooperative IDS. We devised a machine learning-based cooperative IDS that efficiently exploits the historical feedback data to provide the ability of proactive decision making. The machine learning model has been formulated as a Denoising Autoencoder, which is used as a building block for constructing a deep neural network, which allows us to proactively make decisions about suspicious intrusions, even in the absence of complete feedback from the IDSs.

The above contributions are effective in addressing some interesting research gaps in the literature. However, some points still need further study and investigation. The following research avenues could be further explored based on the contributions presented in this thesis and our literature review :

— Although we have paved the road for a generic cloud-based IDS through allowing a low-overhead monitoring on the collected data, and removing irrelevant run-time details from it, the solution still needs to reduce human interaction. Thus, an automated system is required to take the data and automatically abstract and extract robust features from it. For this purpose, deep learning techniques could be the best

candidate to replace the existing detection approaches. The solution can be designed and implemented using different Deep Learning architectures (e.g., Generative Adversarial Networks, Stacked Denosing Autoencoder, Restricted Boltzman Machine, and Variational Autoencoder) for auto-abstraction and extraction of robust features to significantly enhance the detection under heterogeneous, changing and noisy environments. The solution should able not only to accommodate unknown variants of known attacks but also to accommodate unknown variants of unknown attacks.

— Since the direction of the recent research is to automate the process of intrusion detection in single and cooperative settings, we must design automated IDS solutions that are robust against adversarial examples, which are inputs designed by an attacker to fool the machine learning models and make it generate erroneous decisions (e.g., making malware analysis tools unable to detect malicious code). It has been recently seen that Machine Learning models, including Deep Neural Networks, are very vulnerable to adversarial examples. It is easy for an attacker to create "adversarial examples" [178] to fool a machine learning model by simply perpetuating parts of the inputs. Although some work addresses this problem, these solutions are mostly based on adversarial training [179] and are not mature enough to combine the extraction of robust and useful features and preventing the system against adversarial examples. The solution should not only be robust against complex and noisy data but also against adversarial examples.

— The power of most IDSs is largely based on the amount of knowledge that they have about dangerous attacks. In fact, Supervised Machine Learning algorithms such as SVM, used by IDS, are heavily dependent on labeled data to learn how to effectively classify malicious and normal behaviours. However, obtaining malicious behaviours data is challenging and dangerous, especially if we are required to launch real attacks on production systems and put users, applications and systems at risk. Therefore, generative models, that are able to learn the underlying distributions of these complex attacks, in order to generate such sophisticated data and then train machine learning algorithms with it, can improve the detection accuracy. Deep Generative Models such as Generative adversarial Networks (GANS)[180] can be a good candidate to achieve that.

— Moreover, it is important to enable multi-cloud cooperative IDS under a privacy-preserving setting. To this end, generative models will be harnessed to convert the original data available to new data that is similar to the original one logically but not synthetically. This allows us to hide private information in the original data. The quality of the results produced by GANs will allow us to extract useful and robust

features and to share knowledge with other actors without the need to use true data points. The same results can be achieved using generated synthetic data points.

REFERENCES

[1] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A survey of intrusion detection techniques in cloud," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 42–57, 2013.

[2] R. Shea and J. Liu, "Understanding the impact of denial of service attacks on virtual machines," in *Proceedings of the 2012 IEEE 20th International Workshop on Quality of Service*. IEEE Press, 2012, p. 27.

[3] H.-Y. Tsai, M. Siebenhaar, A. Miede, Y. Huang, and R. Steinmetz, "Threat as a service ? : Virtualization's impact on cloud security," *IT Professional Magazine*, vol. 14, no. 1, p. 32, 2012.

[4] Q. Wong, *Salesforce Pushed Silicon Valley into the Cloud*, 1998 (accessed July 11, 2016), http://www.newsfactor.com.

[5] V. Yegneswaran, P. Barford, and S. Jha, "Global intrusion detection in the domino overlay system." in *NDSS*, 2004.

[6] M. Cai, K. Hwang, Y.-K. Kwok, S. Song, and Y. Chen, "Collaborative internet worm containment," *IEEE Security & Privacy*, vol. 3, no. 3, pp. 25–33, 2005.

[7] X. Liu, P. Zhu, Y. Zhang, and K. Chen, "A collaborative intrusion detection mechanism against false data injection attack in advanced metering infrastructure," *IEEE Transactions on Smart Grid*, vol. 6, no. 5, pp. 2435–2443, 2015.

[8] A. Patel, H. Alhussian, J. M. Pedersen, B. Bounabat, J. C. Júnior, and S. Katsikas, "A nifty collaborative intrusion detection and prevention architecture for smart grid ecosystems," *Computers & Security*, vol. 64, pp. 92–109, 2017.

[9] N.-F. Huang, C. Wang, I.-J. Liao, C.-W. Lin, and C.-N. Kao, "An openflow-based collaborative intrusion prevention system for cloud networking," in *Communication Software and Networks (ICCSN), 2015 IEEE International Conference on.* IEEE, 2015, pp. 85–92.

[10] H. Sedjelmaci and S. M. Senouci, "An accurate and efficient collaborative intrusion detection framework to secure vehicular networks," *Computers & Electrical Engineering*, vol. 43, pp. 33–47, 2015.

[11] C. J. Fung and Q. Zhu, "Facid : A trust-based collaborative decision framework for intrusion detection networks," *Ad Hoc Networks*, vol. 53, pp. 17–31, 2016.

[12] A. Abusitta, M. Bellaiche, and M. Dagenais, "A trust-based game theoretical model for cooperative intrusion detection in multi-cloud environments," in *21st Conference on Innovation in Clouds, Internet and Networks (ICIN 2018)*. IEEE, 2018 (to appear soon).

[13] N. Briscoe, "Understanding the osi 7-layer model," *PC Network Advisor*, vol. 120, no. 2, 2000.

[14] A. Josang and R. Ismail, "The beta reputation system," in *Proceedings of the 15th bled electronic commerce conference*, vol. 5, 2002, pp. 2502–2511.

[15] J. H. Dreze and J. Greenberg, "Hedonic coalitions : Optimality and stability," *Econometrica : Journal of the Econometric Society*, pp. 987–1003, 1980.

[16] G. Shafer, "Dempster-shafer theory," *Encyclopedia of artificial intelligence*, pp. 330–331, 1992.

[17] H. Von Stackelberg, *Market structure and equilibrium*. Springer Science & Business Media, 2010.

[18] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders : Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[19] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[20] B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman *et al.*, "Reservoir-when one cloud is not enough," *Computer*, vol. 44, no. 3, pp. 44–51, 2011.

[21] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "How to enhance cloud architectures to enable cross-federation," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. IEEE, 2010, pp. 337–345.

[22] M. Guazzone, C. Anglano, and M. Sereno, "A game-theoretic approach to coalition formation in green cloud federations," in *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*. IEEE, 2014, pp. 618–625.

[23] C. J. Fung, D. Y. Lam, and R. Boutaba, "Revmatch : An efficient and robust decision model for collaborative malware detection," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*. IEEE, 2014, pp. 1–9.

[24] A. M. Lonea, D. E. Popescu, and H. Tianfield, "Detecting ddos attacks in cloud computing environment," *International Journal of Computers Communications & Control*, vol. 8, no. 1, pp. 70–78, 2013.

[25] D. Dittrich, "The 'stacheldraht'distributed denial of service attack tool," 1999.

[26] A. Bakshi and Y. B. Dujodwala, "Securing cloud from ddos attacks using intrusion detection system in virtual machine," in *Communication Software and Networks, 2010. ICCSN'10. Second International Conference on.* IEEE, 2010, pp. 260–264.

[27] S. Gupta and P. Kumar, "Vm profile based optimized network attack pattern detection scheme for ddos attacks in cloud," in *International Symposium on Security in Computing and Communication.* Springer, 2013, pp. 255–261.

[28] I. Gul and M. Hussain, "Distributed cloud intrusion detection model," *International Journal of Advanced Science and Technology*, vol. 34, no. 38, p. 135, 2011.

[29] T. Karnwal, S. Thandapanii, and A. Gnanasekaran, "A filter tree approach to protect cloud computing against xml ddos and http ddos attack," in *Intelligent Informatics.* Springer, 2013, pp. 459–469.

[30] T. Karnwal, T. Sivakumar, and G. Aghila, "A comber approach to protect cloud computing against xml ddos and http ddos attack," in *Electrical, Electronics and Computer Science (SCEECS), 2012 IEEE Students' Conference on.* IEEE, 2012, pp. 1–5.

[31] O. Osanaiye, K.-K. R. Choo, and M. Dlodlo, "Distributed denial of service (ddos) resilience in cloud : review and conceptual cloud ddos mitigation framework," *Journal of Network and Computer Applications*, vol. 67, pp. 147–165, 2016.

[32] M. Ficco and M. Rak, "Stealthy denial of service strategy in cloud computing," *IEEE Transactions on Cloud Computing*, vol. 3, no. 1, pp. 80–94, 2015.

[33] M. Masood, Z. Anwar, S. A. Raza, and M. A. Hur, "Edos armor : a cost effective economic denial of sustainability attack mitigation framework for e-commerce applications in cloud environments," in *Multi Topic Conference (INMIC), 2013 16th International.* IEEE, 2013, pp. 37–42.

[34] A. Koduru, T. Neelakantam *et al.*, "Detection of economic denial of sustainability using time spent on a web page in cloud," in *Cloud Computing in Emerging Markets (CCEM), 2013 IEEE International Conference on.* IEEE, 2013, pp. 1–4.

[35] H. Kwon, T. Kim, S. J. Yu, and H. K. Kim, "Self-similarity based lightweight intrusion detection method for cloud computing," in *Asian Conference on Intelligent Information and Database Systems.* Springer, 2011, pp. 353–362.

[36] F. Palmieri, U. Fiore, and A. Castiglione, "A distributed approach to network anomaly detection based on independent component analysis," *Concurrency and Computation : Practice and Experience*, vol. 26, no. 5, pp. 1113–1129, 2014.

[37] J. Choi, C. Choi, B. Ko, and P. Kim, "A method of ddos attack detection using http packet pattern and rule engine in cloud computing environment," *Soft Computing*, vol. 18, no. 9, pp. 1697–1703, 2014.

[38] N. Jeyanthi and P. Mogankumar, "A virtual firewall mechanism using army nodes to protect cloud infrastructure from ddos attacks," *Cybernetics and Information Technologies*, vol. 14, no. 3, pp. 71–85, 2014.

[39] N. Jeyanthi and N. Iyengar, "Escape-on-sight : an efficient and scalable mechanism for escaping ddos attacks in cloud computing environment," *Cybernetics and Information Technologies*, vol. 13, no. 1, pp. 46–60, 2013.

[40] A. Chonka and J. Abawajy, "Detecting and mitigating hx-dos attacks against cloud web services." in *NBiS*, 2012, pp. 429–434.

[41] A. Chonka, Y. Xiang, W. Zhou, and A. Bonti, "Cloud security defence to protect cloud computing against http-dos and xml-dos attacks," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1097–1107, 2011.

[42] O. A. Wahab, H. Otrok, and A. Mourad, "A dempster–shafer based tit-for-tat strategy to regulate the cooperation in vanet using qos-olsr protocol," *Wireless Personal Communications*, vol. 75, no. 3, pp. 1635–1667, 2014.

[43] N. C. S. N. Iyengar, G. Ganapathy, P. Mogan Kumar, and A. Abraham, "A multi-level thrust filtration defending mechanism against ddos attacks in cloud computing environment," *International Journal of Grid and Utility Computing*, vol. 5, no. 4, pp. 236–248, 2014.

[44] R. A. Michelin, A. F. Zorzo, and C. A. De Rose, "Mitigating dos to authenticated cloud rest apis," in *Internet Technology and Secured Transactions (ICITST), 2014 9th International Conference for*. IEEE, 2014, pp. 106–111.

[45] W. Dou, Q. Chen, and J. Chen, "A confidence-based filtering method for ddos attack defense in cloud environment," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1838–1850, 2013.

[46] D. Meyer, F. Leisch, and K. Hornik, "The support vector machine under test," *Neurocomputing*, vol. 55, no. 1, pp. 169–186, 2003.

[47] P. Negi, A. Mishra, and B. Gupta, "Enhanced cbf packet filtering method to detect ddos attack in cloud computing environment," *arXiv preprint arXiv :1304.7073*, 2013.

[48] P. Shamsolmoali and M. Zareapoor, "Statistical-based filtering system against ddos attacks in cloud computing," in *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*. IEEE, 2014, pp. 1234–1239.

[49] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information theory*, vol. 37, no. 1, pp. 145–151, 1991.

[50] T. Vissers, T. S. Somasundaram, L. Pieters, K. Govindarajan, and P. Hellinckx, "Ddos defense system for web services in a cloud environment," *Future Generation Computer Systems*, vol. 37, pp. 37–45, 2014.

[51] M. I. Ribeiro, "Gaussian probability density functions : Properties and error characterization," *Institute for Systems and Robotics, Lisboa, Portugal*, 2004.

[52] A. K. Marnerides, P. Spachos, P. Chatzimisios, and A. U. Mauthe, "Malware detection in the cloud under ensemble empirical mode decomposition," in *2015 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2015, pp. 82–88.

[53] M. N. Ismail, A. Aborujilah, S. Musa, and A. Shahzad, "Detecting flooding based dos attack in cloud computing environment using covariance matrix approach," in *Proceedings of the 7th international conference on ubiquitous information management and communication*. ACM, 2013, p. 36.

[54] A. Girma, M. Garuba, J. Li, and C. Liu, "Analysis of ddos attacks and an introduction of a hybrid statistical model to detect ddos attacks on cloud computing environment," in *Information Technology-New Generations (ITNG), 2015 12th International Conference on*. IEEE, 2015, pp. 212–217.

[55] M. Zakarya, "Ddos verification and attack packet dropping algorithm in cloud computing," *World Applied Sciences Journal*, vol. 23, no. 11, pp. 1418–1424, 2013.

[56] H. S. Bedi and S. Shiva, "Securing cloud infrastructure against co-resident dos attacks using game theoretic defense mechanisms," in *Proceedings of the international conference on advances in computing, communications and informatics*. ACM, 2012, pp. 463–469.

[57] C. N. Modi, D. R. Patel, A. Patel, and M. Rajarajan, "Integrating signature apriori based network intrusion detection system (nids) in cloud computing," *Procedia Technology*, vol. 6, pp. 905–912, 2012.

[58] B. Cha and J. Kim, "Study of multistage anomaly detection for secured cloud computing resources in future internet," in *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*. IEEE, 2011, pp. 1046–1050.

[59] T. Hastie, R. Tibshirani, and J. Friedman, "Unsupervised learning," in *The elements of statistical learning*. Springer, 2009, pp. 485–585.

[60] M. Ficco, "Security event correlation approach for cloud computing," *International Journal of High Performance Computing and Networking 1*, vol. 7, no. 3, pp. 173–185, 2013.

[61] S. Teng, C. Zheng, H. Zhu, D. Liu, and W. Zhang, "A cooperative intrusion detection model for cloud computing networks," *International Journal of Security and its applications*, vol. 8, no. 3, pp. 107–118, 2014.

[62] C.-C. Lo, C.-C. Huang, and J. Ku, "A cooperative intrusion detection system framework for cloud computing networks," in *Parallel processing workshops (ICPPW), 2010 39th international conference on*. IEEE, 2010, pp. 280–284.

[63] N. D. Man and E.-N. Huh, "A collaborative intrusion detection system framework for cloud computing," in *Proceedings of the International Conference on IT Convergence and Security 2011*. Springer, 2012, pp. 91–109.

[64] D. Singh, D. Patel, B. Borisaniya, and C. Modi, "Collaborative ids framework for cloud," *International Journal of Network Security*, vol. 18, no. 4, pp. 699–709, 2016.

[65] S. Ghribi, "Distributed and cooperative intrusion detection in cloud networks," in *Proceedings of the Doctoral Symposium of the 17th International Middleware Conference*. ACM, 2016, p. 7.

[66] Z. Chiba, N. Abghour, K. Moussaid, M. Rida *et al.*, "A cooperative and hybrid network intrusion detection framework in cloud computing based on snort and optimized back propagation neural network," *Procedia Computer Science*, vol. 83, pp. 1200–1206, 2016.

[67] Á. Dermott, Q. Shi, and K. Kifayat, "Collaborative intrusion detection in federated cloud environments," *Journal of Computer Sciences and Applications*, vol. 3, no. 3A, pp. 10–20, 2015.

[68] M. E. Locasto, J. J. Parekh, A. D. Keromytis, and S. J. Stolfo, "Towards collaborative security and p2p intrusion detection," in *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*. IEEE, 2005, pp. 333–339.

[69] C. G. Cordero, E. Vasilomanolakis, M. Mühlhäuser, and M. Fischer, "Community-based collaborative intrusion detection." in *SecureComm*, 2015, pp. 665–681.

[70] Q. Zhu, C. Fung, R. Boutaba, and T. Basar, "A game-theoretical approach to incentive design in collaborative intrusion detection networks," in *Game Theory for Networks, 2009. GameNets' 09. International Conference on*. IEEE, 2009, pp. 384–392.

[71] ——, "Guidex : A game-theoretic incentive-based mechanism for intrusion detection networks," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 11, pp. 2220–2230, 2012.

[72] C. Fung, Q. Zhu, R. Boutaba, and T. Başar, "Smurfen : A system framework for rule sharing collaborative intrusion detection," in *Proceedings of the 7th International Conference on Network and Services Management*. International Federation for Information Processing, 2011, pp. 248–253.

[73] Q. Zhu, C. Fung, R. Boutaba, and T. Başar, "A game-theoretic approach to rule sharing mechanism in networked intrusion detection systems : Robustness, incentives and security," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. IEEE, 2011, pp. 243–248.

[74] R. Buyya, R. Ranjan, and R. N. Calheiros, "Intercloud : Utility-oriented federation of cloud computing environments for scaling of application services," in *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 2010, pp. 13–31.

[75] M. Fazio, A. Celesti, M. Villari, and A. Puliafito, "How to enhance cloud architectures to enable cross-federation : Towards interoperable storage providers," in *Cloud Engineering (IC2E), 2015 IEEE International Conference on*. IEEE, 2015, pp. 480–486.

[76] I. Goiri, J. Guitart, and J. Torres, "Characterizing cloud federation for enhancing providers' profit," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. IEEE, 2010, pp. 123–130.

[77] A. N. Toosi, R. N. Calheiros, R. K. Thulasiram, and R. Buyya, "Resource provisioning policies to increase iaas provider's profit in a federated cloud environment," in *High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on*. IEEE, 2011, pp. 279–287.

[78] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. IEEE, 2010, pp. 228–235.

[79] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE Transactions on Services Computing*, vol. 5, no. 2, pp. 164–177, 2012.

[80] D. Bruneo, "A stochastic model to investigate data center performance and qos in iaas cloud computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 560–569, 2014.

[81] X. Yang, B. Nasser, M. Surridge, and S. Middleton, "A business-oriented cloud federation model for real-time applications," *Future Generation Computer Systems*, vol. 28, no. 8, pp. 1158–1167, 2012.

[82] M. Salama and A. Shawish, "A qos-oriented inter-cloud federation framework," in *Computer Software and Applications Conference (COMPSAC), 2014 IEEE 38th Annual*. IEEE, 2014, pp. 642–643.

[83] L. Mashayekhy, M. M. Nejad, and D. Grosu, "Cloud federations in the sky : Formation game and mechanism," *IEEE Transactions on Cloud Computing*, vol. 3, no. 1, pp. 14–27, 2015.

[84] M. M. Hassan, B. Song, and E.-N. Huh, "Distributed resource allocation games in horizontal dynamic cloud federation platform," in *High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on*. IEEE, 2011, pp. 822–827.

[85] M. Mihailescu and Y. M. Teo, "Dynamic resource pricing on federated clouds," in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. IEEE Computer Society, 2010, pp. 513–517.

[86] H. Li, C. Wu, Z. Li, and F. C. Lau, "Profit-maximizing virtual machine trading in a federation of selfish clouds," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 25–29.

[87] N. Samaan, "A novel economic sharing model in a federation of selfish cloud providers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 12–21, 2014.

[88] C. Ngo, Y. Demchenko, and C. De Laat, "Toward a dynamic trust establishment approach for multi-provider intercloud environment," in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*. IEEE, 2012, pp. 532–538.

[89] F. Messina, G. Pappalardo, D. Rosaci, C. Santoro, and G. M. Sarné, "A trust model for competitive cloud federations," in *Complex, Intelligent and Software Intensive Systems (CISIS), 2014 Eighth International Conference on*. IEEE, 2014, pp. 469–474.

[90] M. M. Hassan, M. Abdullah-Al-Wadud, A. Almogren, S. Rahman, A. Alelaiwi, A. Alamri, M. Hamid *et al.*, "Qos and trust-aware coalition formation game in data-intensive cloud federations," *Concurrency and Computation : Practice and Experience*, 2015.

[91] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "Towards trustworthy multi-cloud services communities : A trust-based hedonic coalitional game," *IEEE Transactions on Services Computing*, vol. 11, no. 1, pp. 184–201, 2018.

[92] M. Desnoyers and M. R. Dagenais, "The lttng tracer : A low impact performance and behavior monitor for gnu/linux," in *OLS (Ottawa Linux Symposium)*, vol. 2006. Citeseer, 2006, pp. 209–224.

[93] N. Ezzati-Jivan and M. R. Dagenais, "A stateful approach to generate synthetic events from kernel traces," *Advances in Software Engineering*, vol. 2012, p. 6, 2012.

[94] Markus-Go, *BoNeSi - the DDoS Botnet Simulator*, 2015 (accessed July 6, 2016), https://github.com/Markus-Go/bonesi.

[95] A. Bogomolnaia and M. O. Jackson, "The stability of hedonic coalition structures," *Games and Economic Behavior*, vol. 38, no. 2, pp. 201–230, 2002.

[96] C. J. Fung, J. Zhang, I. Aib, and R. Boutaba, "Robust and scalable trust management for collaborative intrusion detection," in *Integrated Network Management, 2009. IM'09. IFIP/IEEE International Symposium on.* IEEE, 2009, pp. 33–40.

[97] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim : a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software : Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[98] A. Perea, "Backward induction versus forward induction reasoning," *Games*, vol. 1, no. 3, pp. 168–188, 2010.

[99] D. Gonzales, J. Kaplan, E. Saltzman, Z. Winkelman, and D. Woods, "Cloud-trust-a security assessment model for infrastructure as a service (iaas) clouds," 2015.

[100] *Control Groups Resource Management*, 2016 (accessed July 6, 2016), https://libvirt.org/cgroups.html.

[101] P. M. L. R. S. D. O. Peter, M. Eva, *Managing system resources on Red Hat Enterprise Linux 6*, 2016 (accessed July 6, 2016), https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html-single/Resource_Management_Guide/.

[102] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "I know you are watching me : Stackelberg-based adaptive intrusion detection strategy for insider attacks in the cloud," in *IEEE International Conference on Web Services (ICWS).* IEEE, 2017, pp. 728–735.

[103] ——, "Optimal load distribution for the detection of vm-based ddos attacks in the cloud," *IEEE Transactions on Services Computing*, 2017.

[104] S. Yu, Y. Tian, S. Guo, and D. O. Wu, "Can we beat ddos attacks in clouds ?" *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 9, pp. 2245–2254, 2014.

[105] G. Somani, A. Johri, M. Taneja, U. Pyne, M. S. Gaur, and D. Sanghi, "Darac : Ddos mitigation using ddos aware resource allocation in cloud," in *International Conference on Information Systems Security.* Springer, 2015, pp. 263–282.

[106] N. Ezzati-Jivan and M. R. Dagenais, "A framework to compute statistics of system parameters from very large trace files," *ACM SIGOPS Operating Systems Review*, vol. 47, no. 1, pp. 43–54, 2013.

[107] K. A. Heller, K. M. Svore, A. D. Keromytis, and S. J. Stolfo, "One class support vector machines for detecting anomalous windows registry accesses," in *Proc. of the workshop on Data Mining for Computer Security*, vol. 9, 2003.

[108] T. Shon and J. Moon, "A hybrid machine learning approach to network anomaly detection," *Information Sciences*, vol. 177, no. 18, pp. 3799–3821, 2007.

[109] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "Misbehavior detection framework for community-based cloud computing," in *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*. IEEE, 2015, pp. 181–188.

[110] O. A. Wahab, A. Mourad, H. Otrok, and J. Bentahar, "Ceap : Svm-based intelligent detection model for clustered vehicular ad hoc networks," *Expert Systems with Applications*, vol. 50, pp. 40–54, 2016.

[111] T. Wood, L. Cherkasova, K. Ozonat, and P. Shenoy, "Profiling and modeling resource usage of virtualized applications," in *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*. Springer-Verlag New York, Inc., 2008, pp. 366–387.

[112] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155–162, 2012.

[113] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on parallel and distributed systems*, vol. 24, no. 6, pp. 1107–1117, 2013.

[114] J. Bognár, *Indefinite inner product spaces*. Springer Science & Business Media, 2012, vol. 78.

[115] L. Auria and R. A. Moro, "Support vector machines (svm) as a technique for solvency analysis," 2008.

[116] A. Konar, U. K. Chakraborty, and P. P. Wang, "Supervised learning on a fuzzy petri net," *Information Sciences*, vol. 172, no. 3, pp. 397–416, 2005.

[117] S. Fine and K. Scheinberg, "Efficient svm training using low-rank kernel representations," *Journal of Machine Learning Research*, vol. 2, no. Dec, pp. 243–264, 2001.

[118] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core vector machines : Fast svm training on very large data sets," *Journal of Machine Learning Research*, vol. 6, no. Apr, pp. 363–392, 2005.

[119] J.-x. Dong, A. Krzyzak, and C. Y. Suen, "Fast svm training algorithm with decomposition on very large data sets," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 4, pp. 603–618, 2005.

[120] S. Kandula, D. Katabi, M. Jacob, and A. Berger, "Botz-4-sale : Surviving organized ddos attacks that mimic flash crowds," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2.* USENIX Association, 2005, pp. 287–300.

[121] *Kernel Virtual Machine*, 2016 (accessed July 6, 2016), http://www.linux-kvm.org/page/Main_Page.

[122] E. A. K. C. P. Hick and J. Polterock, *The CAIDA DDoS Attack 2007 Dataset*, 2007 (accessed July 10, 2016), http://www.caida.org/data/passive/ddos-20070804_dataset.xml.

[123] M. Arlitt and T. Jin, *1998 World Cup Web Site Access Logs*, 1998 (accessed July 10, 2016), http://www.acm.org/sigcomm/ITA/.

[124] S. Bhatia, D. Schmidt, G. Mohay, and A. Tickle, "A framework for generating realistic traffic for distributed denial-of-service attacks and flash events," *Computers & Security*, vol. 40, pp. 95–107, 2014.

[125] *LTTng analyses*, 2016 (accessed July 6, 2016), https://github.com/lttng/lttng-analyses.

[126] S. Maji, A. C. Berg, and J. Malik, "Efficient classification for additive kernel svms," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 66–77, 2013.

[127] R. Kohavi and J. R. Quinlan, "Data mining tasks and methods : Classification : decision-tree discovery," in *Handbook of data mining and knowledge discovery.* Oxford University Press, Inc., 2002, pp. 267–276.

[128] Z. Al-Mousa and Q. Nasir, "cl-cidps : A cloud computing based cooperative intrusion detection and prevention system framework," in *International Conference on Future Network Systems and Security.* Springer, 2015, pp. 181–194.

[129] H. A. Kholidy and F. Baiardi, "Cids : A framework for intrusion detection in cloud systems," in *Information Technology : New Generations (ITNG), 2012 Ninth International Conference on.* IEEE, 2012, pp. 379–385.

[130] D. Ray, *A game-theoretic perspective on coalition formation.* Oxford University Press, 2007.

[131] H. Yahyaoui, "A trust-based game theoretical model for web services collaboration," *Knowledge-Based Systems*, vol. 27, pp. 162–169, 2012.

[132] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "Towards trustworthy multi-cloud services communities : A trust-based hedonic coalitional game," *IEEE Transactions on Services Computing*, 2016.

[133] K. R. Apt and A. Witzel, "A generic approach to coalition formation," *International Game Theory Review*, vol. 11, no. 03, pp. 347–367, 2009.

[134] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé, "Coalition structure generation with worst case guarantees," *Artificial Intelligence*, vol. 111, no. 1-2, pp. 209–238, 1999.

[135] P. K. Sinha, *Distributed operating systems : concepts and design*. PHI Learning Pvt. Ltd., 1998.

[136] M. Wooldridge, *An introduction to multiagent systems*. John Wiley & Sons, 2009.

[137] A. D. Kshemkalyani and M. Singhal, *Distributed computing : principles, algorithms, and systems*. Cambridge University Press, 2011.

[138] B. Yu and M. P. Singh, "An evidential model of distributed reputation management," in *Proceedings of the first international joint conference on Autonomous Agents and Multiagent Systems : Part 1*. ACM, 2002, pp. 294–301.

[139] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "A survey on trust and reputation models for web services : Single, composite, and communities," *Decision Support Systems*, vol. 74, pp. 121–134, 2015.

[140] A. Abusitta, M. Bellaiche, and M. Dagenais, "An svm-based framework for detecting dos attacks in virtualized clouds under changing environment," *Journal of Cloud Computing*, vol. 7, no. 1, p. 9, 2018.

[141] S. Bu, F. R. Yu, X. P. Liu, P. Mason, and H. Tang, "Distributed combined authentication and intrusion detection with data fusion in high-security mobile ad hoc networks," *IEEE transactions on vehicular technology*, vol. 60, no. 3, pp. 1025–1036, 2011.

[142] Y. Liu, Y. L. Sun, S. Liu, and A. C. Kot, "Securing online reputation systems through dempster-shafer theory based trust model," *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY*, vol. 8, no. 6, 2013.

[143] Z. Wei, H. Tang, F. R. Yu, M. Wang, and P. Mason, "Security enhancements for mobile ad hoc networks with trust management using uncertain reasoning," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 9, pp. 4647–4658, 2014.

[144] S. Liu, A. C. Kot, C. Miao, and Y.-L. Theng, "A dempster-shafer theory based witness trustworthiness model," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*. International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 1361–1362.

[145] A. Brandenburger, "Cooperative game theory," *Teaching Materials at New York University*, 2007.

[146] K. Ritzberger *et al.*, "Foundations of non-cooperative game theory," *OUP Catalogue*, 2002.

[147] W. Saad, Z. Han, T. Basar, M. Debbah, and A. Hjorungnes, "Hedonic coalition formation for distributed task allocation among wireless agents," *IEEE Transactions on Mobile Computing*, vol. 10, no. 9, pp. 1327–1344, 2011.

[148] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit : Challenges and opportunities," in *High Performance Computing & Simulation, 2009. HPCS'09. International Conference on.* IEEE, 2009, pp. 1–11.

[149] M. Aazam and E.-N. Huh, "Advance resource reservation and qos based refunding in cloud federation," in *Globecom Workshops (GC Wkshps), 2014.* IEEE, 2014, pp. 139–143.

[150] G. F. Anastasi, E. Carlini, and P. Dazzi, "Smart cloud federation simulations with cloudsim," in *Proceedings of the first ACM workshop on Optimization techniques for resources management in clouds.* ACM, 2013, pp. 9–16.

[151] *Amazon Web Services (AWS) - Cloud Computing Services*, 2018 (accessed may 10, 2018), https://aws.amazon.com/.

[152] S. M. Habib, S. Ries, and M. Muhlhauser, "Cloud computing landscape and research challenges regarding trust and reputation," in *Ubiquitous Intelligence & Computing and 7th International Conference on Autonomic & Trusted Computing (UIC/ATC), 2010 7th International Conference on.* IEEE, 2010, pp. 410–415.

[153] W. Fan, S. Yang, and J. Pei, "A novel two-stage model for cloud service trustworthiness evaluation," *Expert Systems*, vol. 31, no. 2, pp. 136–153, 2014.

[154] M. Alhamad, T. Dillon, and E. Chang, "Sla-based trust model for cloud computing," in *Network-Based Information Systems (NBiS), 2010 13th International Conference on.* IEEE, 2010, pp. 321–324.

[155] S. G. Grivas, T. U. Kumar, and H. Wache, "Cloud broker : Bringing intelligence into the cloud," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on.* IEEE, 2010, pp. 544–545.

[156] B. Rashidi, C. Fung, and E. Bertino, "A collaborative ddos defence framework using network function virtualization," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 10, pp. 2483–2497, 2017.

[157] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[158] G. E. Hinton, "Learning multiple layers of representation," *Trends in cognitive sciences*, vol. 11, no. 10, pp. 428–434, 2007.

[159] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[160] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[161] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, 2007, pp. 153–160.

[162] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[163] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in *Machine Learning and Applications (ICMLA), 2016 15th IEEE International Conference on*. IEEE, 2016, pp. 195–200.

[164] J. Kim, N. Shin, S. Y. Jo, and S. H. Kim, "Method of intrusion detection using deep neural network," in *Big Data and Smart Computing (BigComp), 2017 IEEE International Conference on*. IEEE, 2017, pp. 313–316.

[165] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016, pp. 21–26.

[166] S. Potluri and C. Diedrich, "Accelerated deep neural networks for enhanced intrusion detection system," in *Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on*. IEEE, 2016, pp. 1–8.

[167] C. G. Cordero, S. Hauke, M. Mühlhäuser, and M. Fischer, "Analyzing flow-based anomaly intrusion detection using replicator neural networks," in *Privacy, Security and Trust (PST), 2016 14th Annual Conference on*. IEEE, 2016, pp. 317–324.

[168] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Wireless Networks and Mobile Communications (WINCOM), 2016 International Conference on*. IEEE, 2016, pp. 258–263.

[169] C.-Y. Liou, W.-C. Cheng, J.-W. Liou, and D.-R. Liou, "Autoencoder for words," *Neurocomputing*, vol. 139, pp. 84–96, 2014.

[170] A. J. Bell and T. J. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural computation*, vol. 7, no. 6, pp. 1129–1159, 1995.

[171] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[172] S. Haykin and N. Network, "A comprehensive foundation," *Neural networks*, vol. 2, no. 2004, p. 41, 2004.

[173] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[174] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 473–480.

[175] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models : a methodology review," *Journal of biomedical informatics*, vol. 35, no. 5-6, pp. 352–359, 2002.

[176] *KDD Cup 1999 Data*, 2018 (accessed may 10, 2018), http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[177] A. Zee, "Emergence of grandmother memory in feed forward networks : Learning with noise and forgetfulness," *Connectionist models and their implications : Readings from cognitive science*, p. 309, 1988.

[178] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv :1607.02533*, 2016.

[179] N. Carlini and D. Wagner, "Audio adversarial examples : Targeted attacks on speech-to-text," *arXiv preprint arXiv :1801.01944*, 2018.

[180] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.